Università degli Studi di Padova

Department of Information Engineering

Computer Science Engineering

Methods for Learning Ensembles of Deep Neural Networks

Author:
Gianluca Maguolo

Supervisor:
Prof. Loris Nanni

September 30, 2021

Contents

Abstract

In this thesis, I study methods to create ensembles of deep neural networks. Ensembles of deep neural networks are sets of deep networks whose outputs are aggregated together to obtain a single result. The idea behind ensembles is that the overall prediction of multiple networks will be better than the ones of the single networks.

I introduce many methods used in the literature to create ensembles and I divide them into different categories depending on how the classifiers are forced to learn different features of the input data. Our experiments show that using deep networks that are different from each other in their architectures, in their training or in their image preprocessing allows to reach better performances than those that are reached by baseline ensembles of well-performing but similar networks.

Chapter 1

Introduction

Convolutional Neural Networks (CNNs) are one of the most effective tools in modern computer vision [1]. Most state of the art performances on tasks like image classification [2], image segmentation [3] and object recognition [4] are currently achieved using CNNs. Experiments show that the performance of CNNs can be improved by training a large number of models and averaging their results. Such a set of neural networks is called an ensemble [5, 6, 7, 8]. The idea behind averaging the results of multiple classifiers is that multiple opinions are better than one. Besides, in this way every network can specialize on a specific part of the data. Finally, a method to aggregate the decision of the different networks is required.

Ensembles are very old tools in machine learning and date back to much earlier than the rise of modern deep learning. The use of ensembles for supervised learning has been explored by Tukey [5], who created an ensemble of two linear regression models. He proposed to fit the first linear regression model to the original data and the second one on the residuals. Dasarathy and Sheela [6] used component classifiers trained from different categories to create a classification tool, that managed to improve the accuracy of identification systems. In 1990, Hansen and Salamon [7] proved that ensembles of neural networks could decrease the generalization error of neural networks. In the same year, Schapire [8] described how to outperform one strong learner using multiple weak learners in a probably approximately correct (PAC) sense. This work was the seed for the forthcoming AdaBoost (adaptive boosting) algorithm. In the next years, the difference between the performance of single models and ensembles was further investigated and there was growing evidence that ensembles outperformed single models [9, 10, 11, 12, 13, 14, 15]. Besides, the diversity of the predictions of the models in the ensemble seemed to be a key for their effectiveness [11, 16]. However, it has been shown that using multiple strong models yields a better results than creating weaker models that are forced to be diverse in their predictions [17]. Hence, diversity in the ensembles should not be the only requirement. The performance of the different models in an ensemble can be combined in multiple ways that can be divided into two main categories: learned and not learned. In the former case, the rule for combining the results is trained using training or validation data. For example, the rule might be a weighted average of the single softmax outputs of the classifiers. In the latter case, the choices might be the average of the softmax output or the class that represent the mode of the model predictions. The drawback of ensemble methods is the fact that they require more computation than a single model and they have more memory requirements.

An early application of CNN ensembles consisted in the prediction of Go moves made by professional players [18]. That ensemble of CNN managed to reach a performance that, at that time, was the state of the art, and managed to outperform every single model. The widespread application of ensemble techniques to CNNs is more recent and follows the popularity of deep learning for computer vision problems. In the ImageNet Large-Scale Visual Recognition Challenge 2012, the winner was an ensemble containing multiple networks with the same architecture, that is called Alexnet. In the ImageNet Large-Scale Visual Recognition Challenge 2014 the winner (GoogLeNet) and the runner-up (VGG) methods consisted of ensembles of multiple architectures to maximize their performance. These ensembles were made by simple retraining the same architecture multiple times with different initializations and using the stochasticity of the training process. In the following years, different methods for CNN ensemble learning were proposed. Classical methods like model averaging, bagging, Adaboost and stacking were applied to convolutional networks to try to get more diverse models in the ensemble.

In general, there are many ways to create a diverse ensemble. The easiest one consists in using different models and architectures. An other one is to train the same model on different training sets. This can be done by randomly splitting the training data into subsets, or by using different data augmentation techniques in every training. For example, Nanni et al. [19] trained an ensemble of networks using different a preprocessing for every network. An other method relies on using different training processes for every network, that leads to different minimums of the optimization landscape even when the models and the datasets are the same. Huang et al. [20] created an ensemble of networks by training the same network for many epochs with an oscillating learning rate and saved the network at different points of the training process. In many computer vision tasks, the datasets are not very large, hence old handcrafted features might be competitive with CNNs. It has been shown in [19] that an ensemble of CNNs and SVMs trained on handcrafted features outperform the ensemble of CNNs. This is probably due to the fact that handcrafted features are deterministic algorithms that search for relevant features of an input and are less prone to overfitting. However, they have less representation power than CNNs and do not use transfer learning, hence the diversity in such an ensemble is very large. In this thesis, I present a large number of effective methods to create and prune ensembles of CNNs. In Chapter 2 I present some state of the art methods proposed in the literature. I discuss the methods to vary the models, the training process, the training sets and the fusion rules that are currently used in the literature. In Chapter 3 I summarize the ensemble methods that are more common in the literature, divided by field of application. In Chapter 4 I focus on methods to create ensembles using different data augmentation and preprocessing techniques. In Chapter 5 I show how to create ensemble by varying the activation functions of a given network architecture. In Chapter 6, I show how training a model with different training algorithms can produce diverse results that can be used to form an ensemble. In Chapter 7 I introduce Siamese Neural Networks and I prove that they can be used to create dissimilarity spaces to train an ensemble of classifiers on the extracted data. In Chapter 8, I present a work of general interest on the ability of neural networks to generalize well among datasets investigating the existing literature in Covid-19 detection from X-Ray images. In Chapter 9 I draw some conclusions based on the results of the previous chapters.

Chapter 2

Related Work

An ensemble of CNNs can be created in several ways. The heuristics approach to build an ensemble, which is also supported by the experiments that we shall do, is that the key to create a well performing ensemble consists in using a large number of networks, whose performance is high and whose predictions are as more independent as possible. I classify the strategies to do that in the following way: training networks with different architectures, using different training sets to train the same network architecture, using different training algorithms and using different rules to combine the networks. I shall now describe the different approaches.

It is worth mentioning that the literature on neural network ensembles often lacks enough experimentation in terms of datasets and fair comparisons from a scientific point of view. Most of the papers that I shall mention in this chapter and in the following ones only test basic neural networks on a dataset which is not highly competitive and show that the ensemble outperforms the stand-alone model, which is not particularly surprising given the long history of ensembles outperforming stand-alone models. Those papers are introduced as a reference to give a snapshot of the history in this topic, since many of the techniques that will be introduced in this thesis fall into categories that have already been considered in the literature, although they were not intensively tested. I shall briefly mention those works and spend more words on the minority that contain more extensive experiments or that established state-of-the-art performances on competitive datasets.

2.1 Ensembles of Different Models

The most intuitive way to create a diverse ensemble is to train different models on the whole dataset. However, it is hard to find different models whose performance is similar and using a large number of models might lead to a performance which is very close to the one of the best model. On the other hand, the predictions of high performing models might be strongly correlated and their ensemble might be as good as one single model. Hence, the challenge consists in finding semi-independent models with high accuracy.

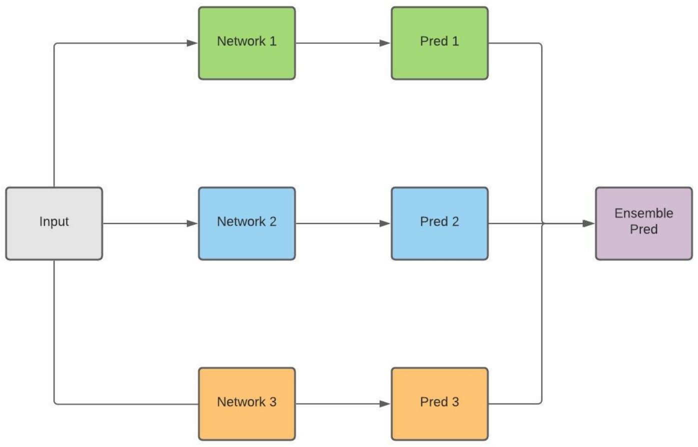


Figure 2.1: Ensemble of different models

The most frequent method consists in using different CNN architectures.

Paul et al. [21] used an ensemble of CNNs to established a new state-of-the-art on the National Lung Screening Trial (NLST), pushing it from 0.87 to 0.96 AUC. Their aim was to predict nodule malignancy from computed tomography. Their approach consisted in a small ensemble of five classifiers, three of which are simple CNN architectures trained from scratch and the remaining two are a pretrained VGG16 and a SVM trained on 219 features extracted from the input image specifically designed to create a radiomics model. The result of the best custom CNN of 0.87 was already above the state-of-the-art AUC, while the transfer learning and the radiomics models did not reach a 0.8 AUC. However, the ensemble of the five models could outperform the ensemble of the custom CNNs with a margin of 0.02 AUC, showing that adding diverse features, some of which are handcrafted, improves the results of the ensemble even when they perform much worse than the other classifiers in the ensemble.

However, in most papers introduced here the authors only fine-tune or train from scratch well-known architectures, average the results and show that this result outperforms the stand-alone networks. For example, in [22] Kassani et al. use an ensemble of VGG19 [23], MobileNet [24] and DenseNet [2] to classify histopathological biopsy and the ensemble of the three networks consistently outperform every stand-alone network on four different datasets. Qummar et al. [25] proposed an ensemble containing ResNet50 [26], Inception v3 [27], Xception [28], DenseNet121 and DenseNet169 [2] to detect diabetic retinopathy. They do not report the results of the single net-

Ref	Field	Description	Baseline	Improvement
[21]	Lung Nodule Detection	Ensemble of custom CNNs, pretrained VGG and SVM	state-of-the-art	0.87 to 0.96 AUC
[22]	4 biopsy classification datasets	Ensemble of VGG19, MobileNet and DenseNet	Best of stand-alone	0.875 to 0.927 avg accuracy
[25]	Diabetic retinopathy	Ensemble of ResNet50, Inveption v3, Xception, Densenet121 and DenseNet169	state-of-the-art	41.6 to 53.7 F1-score
[29]	Facial expression recognition	Ensemble of custom networks	Best stand-alone	0.624 to 0.65 accuracy
[31]	Medical image retrieval	Ensemble of SVMs trained on features extracted from Alexnet and Googlenet	Best stand-alone	0.808 to 0.825 accuracy
[34]	Food recognition	Concatenation of Alexnet, GoogleNet and ResNet50 followed by a dense layer	Best stand-alone	0.663 to 0.728 accuracy
[35]	Image Retrieval	20 Alexnet and 20 NiN	Ensemble of single architectures	0.523 to 0.5236 mean average precision
[37]	Biomedical Image Classification	Ensemble of different architectures and handcrafted features	state-of-the-art	0.92 to 0.952 F1-score
[39]	Lung Nodule Detection	Eight CNNs of Different Networks	Best stand-alone	0.817 to 0.840 accuracy

works. Liu et al. [29] created an ensemble of three different CNNs that they proposed in their paper and averaged their results. Their ensemble reached an accuracy better than the best stand-alone model on the FER2013 dataset [30]. Kumar et al. [31] proposed an ensemble of AlexNet and GoogleNet [32] pretrained on ImageNet and finetuned on the ImageCLEF 2016 collection dataset [33]. Then they used the features extracted by the last fully connected layers of those networks to train a SVM. Their method improves the level of the CNN baselines and is competitive with the state-of-the-art methods of that time. Pandey et al. [34] proposed FoodNet, an ensemble of AlexNet, GoogleNet and ResNet50, finetuned to recognize food images. After that, their output features are concatenated and passed to a fully connected layer and a softmax classifier. Huang et al. [35] created an ensemble of 20 AlexNet and 20 NIN [36] for image retrieval. Their output is then summed to create a unique feature, using a weight of 0.5 for AlexNet and 1 for NIN, since it has a better performance. The proposed method outperforms both the ensembles of 20 AlexNets and 20 NINs. Nanni et al. [37] used an ensemble of AlexNet, GoogleNet, VGG16, VGG19, ResNet50, Inception v3 and Inception-Resnet-v2 [38]. Besides, they extracted handcrated features and trained SVMs on those features and on the ones extracted by the last convolutional layer of the networks. They managed to reach performances that were the state-of-the-art on several bioimage datasets when the paper was published. Zhang et al. [39] used an ensemble of eigth CNNs with different architectures to classify benign and malignant pulmonary nodules using CT images. They showed that the correlation among deep learning classifiers is much higher than the one among classical classifiers. Besides, they showed that their ensemble performed better than every single network.

2.2 Ensembles Using Different Training Sets

Using different training sets to train a classifier is an effective way to learn semi-independent classifiers. This can be done in several ways. A classical way to do this is bagging [40, 41, 42]. Bagging consists in generating m training sets of size n out of a training set of size n by randomly choosing the samples with uniform probability and with replacement. Then, the same model is trained on every training set. Kim et al. [43] proposed an approach based on bagging to train three different CNNs for vehicle type classification and Dong et al. [44] used bagging and CNNs for short-term load forecasting in smart grid, obtaining an improvement from 33.47 to 28.51 in mean absolute percentage error (MAPE). Guo et al. [45] used eight different datasets to train eight different networks for object detection. They obtained the datasets by combining existing datasets in different ways. With this simple approach they managaed to largely outperform the single models and to be close to the state-of-the-art on a very competitive dataset like COCO 2012. Savelli et al. [46] proposed an ensemble of CNNs for micro-calcification detection on full-field digital mammograms and microaneurysm detection on ocular fundus images. They trained five CNNs on image patches of different size, so that every model learns to recognize a different context. They showed that this approach improved the single model that was only trained on the full images.

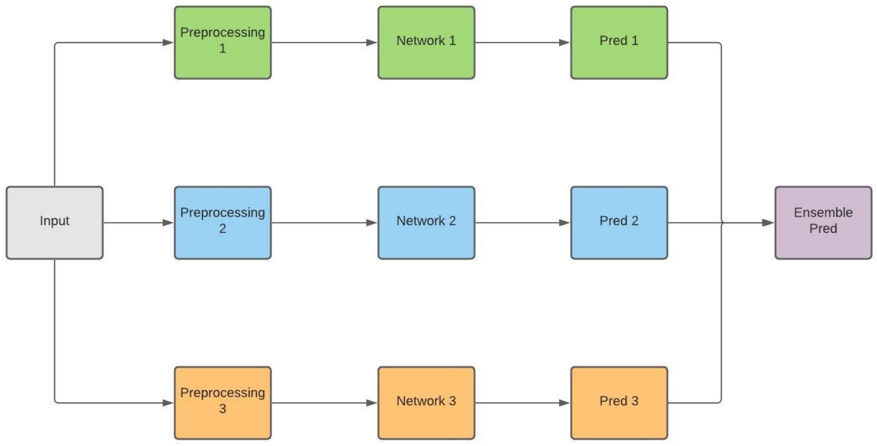


Figure 2.2: Ensemble of different preprocessings

In order to train the same network on different training sets, the same model can be trained on the whole training set, but using different data augmentation techniques in every dataset. Chen et al. [47] created an ensemble to classify hyperspectral images. Their idea consisted in training multiple models trained on different image bands. They randomly sampled

Ref	Field	Description	Baseline	Improvement
[43]	Vehicle type classification	Ensemble of CNNs using bagging	Large CNN	0.9760 to 0.9784 accuracy
[44]	Smart grid forecast	Ensemble of CNNs using bagging	Best of stand-alone	33.47 to 28.51 avg MAPE
[45]	COCO 2012 object detection	Ensemble of eight networks trained on different combinations of different training datasets	state-of-the-art	70.7 to 70.1 mean average precision
[46]	Microcalcification detection	Ensemble of patches of various sizes	Stand-alone on full image	0.78 to 0.825 avergage accuracy
[47]	Hyperspectral image classification	Ensemble of CNNs on subset of channels	Best stand-alone trained on all channels	0.919 to 0.935 average accuracy
[48]	Food Facial expression recognition	Ensemble of different image patches	VGG-Face	0.736 to 0.767 average accuracy

3 of the 27 bands of a hyperspectral image dataset multiple times, and each time they trained a deep network on those three bands. Hence, they created an ensemble of images with different bands. They showed that this ensemble consistently outperformed a single deep CNN on a large number of datasets, which gives statistical value to their findings. However, they did not test their ensemble against a baseline ensemble model, so the result of their study is still not relevant.

The same holds for other papers on this topic, most of which do not even have the merit to have extensively tested their method on multiple datasets. Fan et al. [48] trained an ensemble of CNNs models for facial expression recognition using different regions of the face for training every network. Hence, every network learns different features of the dataset. In order to make the test consistent with the training, the same bands are used as inputs in the relative networks at inference time. In general, this kind of approach can be described as creating different versions of the dataset using different preprocessings on the training and test sets (in the case of Chen et al. [47], this preprocessing is just a projection that deletes some of the channels).

2.3 Ensembles Using Different Training Algorithms

The training algorithms used for convolutional neural networks have stochastic trajectories and work on stochastic batches of data. Hence, the same network trained multiple times might differ at the end of the training process. However, in order to make the final models more different from each other, they can be trained using different training algorithms. For example, Gan et al. [49] created an ensemble for facial expression recognition using soft-label perturbation, propagating different losses for different samples. Antiov et al. [50] used different network initializations to train different networks for gender predictions from face images.

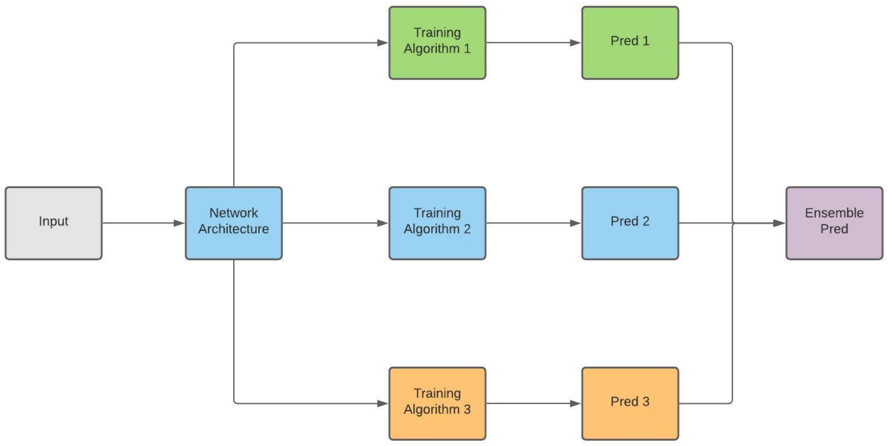


Figure 2.3: Ensemble of different training algorithms

2.4 Decision Rules in Neural Ensembles

An ensemble is a collection of neural networks whose output is combined to get a single response. The way to combine the outputs of the networks is a major decision in ensemble design. A simple rule is majority voting. Majority voting consists in choosing the output which is chosen by the relative majority of the networks in the ensemble [51, 52, 53, 54]. A different way to fuse the outputs is to average the softmax output of the networks. This method is very popular and is the most common choice in the literature [55, 11, 12].

More complicated rules have been proposed. Lumini et al. [56] compute a learnt weighted average of the sotmax output. A special case of decision rule is Adaboost. Adaboost [57] is a decision rule that consists in training a sequence of classifiers, each of which focused on the samples misclassified by the previous ones. In particular, let $D = \{d_i\}$ be a set of weights associated to the N samples of the training set. Every $d_i = \frac{1}{N}$ at the beginning. Then, every model is trained using D as weights of the loss function. These weights are always updated due to the following formula:

$$d_i^{m+1} = d_i^m \exp\left(-\alpha \frac{K-1}{K} \log\left(p_i^m\right)\right), \tag{2.1}$$

where K is the number of classes, α is a learning rate and p_i is the probability assigned to the correct class of the i-th sample by the m-th classifier. It is clear that if the sample is classified correctly, then the next weight will be very close to one, while if it is wrongly classified is much larger than one.

Then $h_k^m(x)$ is defined as

$$h_k^m(x) = (K-1)\left(\log\left(p_k^m(x)\right) - \frac{1}{K}\sum_{\tilde{k}=1}^{m}\log\left(p_{\tilde{k}}^m(x)\right)\right) \tag{2.2}$$

and the output of the model is defined as

$$C(x) = \arg\max_k \sum_{m=1}^{M} h_k^m(x) \tag{2.3}$$

The definition of the algorithms might look unintuitive, however it is based on iterative training error minimization. This algorithm gained popularity before CNNs and it was used to boost the performance of weak classifiers, however it was recently applied to CNNs. Tahernaki et al. [58] trained a sequence of CNNs for classification, iteratively retraining the same network using the weights of the previous step as initialization. They applied this technique to imbalanced classification, with the idea that the first classifiers would have learned the most frequent classes and that the classes in the tail of the distribution can be learnt by the last classifiers. A similar approach was used by Potes et al. [59] for heart sound recognition. They extracted an image from the sound by mapping it into the frequency domain and then used an ensemble of CNNs for the classification. Gao et al. [60] used Adaboost for head regions proposal for video surveillance. They used CNNs for feature extraction and then they train an ensemble of SVMs for region proposal. Chen et al. [61] used Adaboost to train an ensemble of SVMs on features extracted from a CNN for real-time vehicle classification. Zhou et al. [62] proposed an algorithm based on Adaboost for online object tracking. They trained deep networks for classification on the iteratively acquired data during the online training. This allows online training for deep networks without the need to retrain the models on the whole dataset.

2.5 Diversity-Based Ensembles

An important issue in ensemble learning algorithms is how to decorrelate the predicitons of the base models, i.e: letting the networks understand different features of the input samples space without decreasing their individual accuracies. This is called ensemble diversity in the literature [10, 63].

Evolutionary approaches were very popular to choose the networks whose predictions had the largest disagreement [64, 65, 66, 67]. However, those algorithms are very slow. In [10] the models were trained with cross-validation datasets, which means that the models were trained on other datasets to see which ones had different outputs on those dataset, so that they could be selected to be in the ensemble. They used a diversity metric to balance the covariance and the performance of the models. The trade-off between bias and covariance led researchers to create ensembles whose overall prediction had the minimum variance, so that it could be retained across different datasets [63].

Negative correlation learning (NCL) [68, 69, 70, 71] consists in training all the models in an ensemble using an indipendent loss for every model, as the usual crossentropy, and a shared loss that calculates the correlation among the networks in an ensemble [63]. A regularized version of NCL techniques was introduced in [72], and managed to reduce the overfitting on outliers. However, these models are not compared with other ensembles, so the findings in the paper cannot be fully trusted.

Kariyappa et al. [73] proposed gradient alignment loss (GAL) to encourage diversity between different models in an ensemble and applied this technique as defense against adversarial examples. The idea consists in finding a good defence against black-box adversarial attacks, hence the attacker does not know the weights of the model, he or she only knows the architecture and the training dataset. By training multiple models discouraging gradient alignment, the authors made it less likely that the same perturbation affected multiple models in the ensemble. This method is tested against a baseline ensemble and it is shown to be effective against black-box attacks. Dvornik et al. [74] used an opposite approach for few-shot classification, as they trained multiple networks using a Kulback-Leiber loss to encourage the newtorks to have similar predictions. They showed that this allowed the single network to have a better performance and, surprisingly, also their ensemble seemed to benefit from this approach. However, they acknowledge that this method works for small ensembles, as larger ones contain networks that are too similar to each other to provide a real contribution.

There are many measures of diversity in ensembles [9]. One of them is the Q-statistics [75]. Given two classifiers i and j, let $N_{1,1}$ be the number of test samples correctly classified by both classifiers, $N_{0,0}$ be the number of samples wrongly classified by both, and $N_{0,1}$ and $N_{1,0}$ be the ones misclassified only by, respectively, the first and the second classifier.The Q-statistics is defined as

$$Q_{i,j} = \frac{N_{1,1}N_{0,0} - N_{1,0}N_{0,1}}{N_{1,1}N_{0,0} + N_{1,0}N_{0,1}}. \tag{2.4}$$

Another measure based on disagreement is the correlation coefficient [76]

$$\phi = \frac{N_{1,1}N_{0,0} - N_{1,0}N_{0,1}}{\sqrt{(N_{1,1} + N_{0,1}) + (N_{1,1} + N_{1,0}) + (N_{0,0} + N_{0,1}) + (N_{0,0} + N_{1,0})}}. \tag{2.5}$$

It can be proved that $|\phi| \leq Q$. The two measures are very similar, but Q is much simpler to calculate.

The disagreement measure [77, 78] of two classifiers is defined as

$$Q_{i,j} = \frac{N_{1,0} + N_{0,1}}{N_{1,1} + N_{0,0} + N_{1,0} + N_{0,1}}. \tag{2.6}$$

and it calcultates the ratio between the number of samples classified in a different way and the total number of samples. A very similar measure is the double-fault measure [79]

$$Q_{i,j} = \frac{N_{0,0}}{N_{1,1} + N_{0,0} + N_{1,0} + N_{0,1}}. \tag{2.7}$$

that calculates how many samples are misclassified by both classifiers.

Li et. al [80] investigated the effect of diversity among the models in an ensemble and they find that a larger diversity leads to a smaller hypothesis space and to a better generalization. They provide a detailed theoretical analysis on the relation between diversity and reduction of the error of the ensemble, leading to Diversity Regularized Ensemble Pruning, which is an algorithm to find the best subset of classifiers to create an ensemble. However, their method is applied to standard classifiers and not to deep networks.

Cavalcanti et al. [81] also proposed a method for selecting a subset of trained networks to create a diverse ensemble. They also acknowledge the fact that the diversity between the networks is important to achieve a high performance and use multiple methods to evaluate the correlations among the trained networks. However, they used a genetic algorithm to decide which networks to include in the ensemble and its performance is evaluated on a validation set. They applied their method to a variety of datasets and showed that their ensemble is competitive with the best performing stand-alone network on every dataset. However, they also did not apply this method to deep networks.

2.6 Fast Ensembles

During the training of a neural network, its weights continue to change even after the moment when a local minimum is reached. This is due to the noise of the training on batches, the momentum of the optimization method and a possibily large learning rate. Hence, during training, one can get very different networks whose performance is quite similar and that can be used as an ensemble. This technique was called snapshot ensemble in [20] and has the advantage that it requires a small amount of training time, with respect to training multiple networks from scratch. They used a cyclic learning rate defined by

$$\lambda(t) = \frac{\lambda_0}{2}\left(\cos\left(\frac{\pi \mathrm{mod}\left(t-1, \left[\frac{T}{M}\right]\right)}{\left[\frac{T}{M}\right]}\right) + 1\right), \tag{2.8}$$

Ref	Field	Description	Baseline	Improvement
[20]	Cifar-100	Snapshot ensemble with variable learning rate	Single model	0.257 to 0.228 error rate
[82]	Fault diagnosis	Snapshot ensemble averaging learning rates	Single Model	0.966 to 0.955 avg accuracy
[45]	Signature verification	Snapshot ensembles with multiple losses	state-of-the-art	0.961 to 0.613 EER
[83]	Cifar-100	Snapshot ensemble with Adaboost	Standard snapshot ensemble	0.731 to 0.745 avg accuracy
[84]	Cervical cells classification	Snapshot ensemble, transfer learning	Single CNN	0.648 to 0.655 accuracy
[85]	Cifar-100	Ensemble by model perturbation	Single model	0.648 to 0.647 average accuracy

where T is the training time, M are the number of cycles, t is the current iteration and λ_0 is the initial learning rate. This function reaches its maximum when the argument of the cosine is 0 and its minimum when it approaches π. Hence, at the end of every cycle, a new network is added to the ensemble. After that, the sudden increase of the learning rate takes the optimization process in an other point of the optimization landscape and then it hopefully converges to a new local minimum. They show that their method outperform the stand-alone baseline. Of course, the ensemble method needs more inference time and memory, but the same training time.

Wen et al. [82] also proposed a similar method, but they averaged the learning rate of Equation 2.8 with a positive number λ to ensure the fact that the learning rate never reaches zero. They showed that this method improves the baseline results of stand-alone networks in fault diagnosis in three different datasets.

Masoudnia et al. [86] trained a network in a multi-loss framework. In their paper they trained a network several times, always starting from the end point of the previous training, and every time using a different loss. They used all the end points in an ensemble for offline signature verification and managed to show that this method outperformed every single network trained with only one loss and also their ensemble. They improve the state-of-the-art from 0.961 to 0.613 Equal Error Rate (EER)

Zhang et al. [83] used a discrete learning rate decay schedule that consisted in multiplying it by 0.1 every time the validation loss got stucked and was reset to the original value at the end of every training round. They also used Adaboost to train the different networks, but used different decision rules based on different meta learners trained on the validation set. They showed that their method obtains better results than simple snapshot ensembling, but they did not compare ensembles of the same size. They tested their method on the competitive CIFAR-10 and CIFAR-100 image datasets [87] and on the IMDB dataset for sentiment analysis [88]. However, it is worth noticing that they extract two validation sets that they use for training the ensemble that they proposed, but it is not mentioned if they are somehow used for training the baseline models.

Chen et al. [84] addressed the problem of transfer learning in snapshot ensembles, as long training rounds might lead to overfitting in small datasets. For example, they applied snapshot ensembling to cervical cells classification, using a dataset that has a few hundred images. Hence, they added a retraining of the first layers of their CNN on the dataset that was initially used for transfer learning, in their case ImageNet.

An other way to create even faster ensembles is Parameter Ensembling by Perurbation (PEP) [85]. Also in this case, only one network is trained. However, the ensemble is created by perturbing the weights of the final network with an additive Gaussian random noise. The authors show that this method allows to outperform the original network using the right amount of noise, that should be determined using a validation set.

Chapter 3

Deep Ensembles Applications

In this chapter, I review the relevant literature dealing with the different fields where deep ensembles have been and are nowadays applied. In these papers, comparisons are always done between single models and ensembles, with very few exceptions. Besides, the reported results are usually not compared with the state-of-the-art if they are tested on a competitive dataset, while other authors use a dataset they they own or introduce, so no relevant comparisions with other methods are reported. I review these papers since they stil give a relevant contribution in showing how deep learning and ensembles can be used in different fields.

3.1 CIFAR classification

One of the most competitive fields in modern computer vision is the classification of general images, since images can be very different from each other. An example of this kind of task is the well-known CIFAR [87] dataset.

Chen et al. [89] created an ensemble of networks whose inputs are the different layers of a common neural network. They took a deep network, and then they used one of hidden layers to train a new network with a different architecture. Then, they trained a different network starting from one of the successive layers, and so on. In this way, they created an ensemble of networks. At each iteration, they allowed the weights of the original network to change, and in this way they managed to iteratively adjust the weights of all its layers. They also used a loss function that encouraged the network to predict rare classes. Using this strategy, they managed to outperform many previous approaches on CIFAR-10 and CIFAR-100.

Fielding et al. [90] proposed Swarm Optimised Block Architecture Ensembles, a method to automatically design deep CNN ensembles with a shared weight repository to avoid storing the same information multiple times. They trained their models using particle swarm optimisation. They showed that their method managed to outperform the best single network without significant memory overhead. However, the computation time was much larger due to the size of the ensemble, which was 32 networks.

Huang et al. [35] proposed an ensemble of AlexNet and Network-In-Network (NIN) to create image features for image retrieval. They extracted image features by summing the final hidden layer of the networks in the ensemble. These features were used to classify the images of the CIFAR-10 and CIFAR-100 datasets and the authors showed that they managed to outperform an ensemble of 20 NIN architectures independently trained.

Lee et al. proposed [91] Stochastic Multiple Choice Learning, a modification of SGD that allows an ensemble of networks to train simultaneously and at the same time and an oracle is trained to pick the correct output based on the predictions of the network. They reach an accuracy larger than 95% on CIFAR-10 using only 6 small networks.

Dutt et al. [92] proposed a method to extract an ensemble using different branches of a neural network. They created these branches so that each one of them was expert on a specific subset of labels. They showed that they reached an accuracy of 97.08% on CIFAR-10 and 84.32% on CIFAR-100.

3.2 Face recognition

One of the most relevant applications of computer vision is face recognition. Since this application is crucial in many fields, very large datasets are available. For example, the Labeled Faces in the Wild (LFW) dataset [93], the CASIA-WebFace database [94] and the MS-Celeb-1M [95].

One the most famous methods for face recognition is DeepID2 [96]. It consists of an ensemble of CNNs trained and evaluated on different patches. Then, the output of those CNNs are concatenated and their dimension is reduced using PCA. After that, they are classified using a Joint Bayesian model [97]. They report a 98.92% acuracy on LWF.

Ding et al. [98] focused on the problem of robustness in face recognition, since multimedia applications like the ones they were studying usually present large variance in pose, expression and illumination. They trained multiple CNNs on different patches of the input images and used them to create a feature vector by concatenating the final layer of every CNN. Then, they used Stacked Autoencoders to reduce the dimension of the feature vector. Finally, the feature vector obtained in this way was then classified using a Joint Bayesian model [97]. This method is very similar to DeepID2, but it only uses eight networks and replaces the PCA with the autoencoders. They report a 99.02% accuracy on LFW.

Masi et al. [99] proposed an ensemble of five networks to solve the problem of pose invariance in face recognition by training each model with images of pose with different angles. At inference time, they used a pose estimator to decide which network to use for classification. They proved that their method was very effective, but it has the limitations that it really benefits from having images of the same subject with multiple poses in the training set.

Wu et al. [100] proposed a very effective and well tested method based on a deep convolutional network and on nearest neighbour for low-shot face recognition. They trained and evaluated their model on a large dataset that contained multiple samples per person, then they used nearest neighbours to classify the images representing those people that appeared only once in the training set. They used the two models as an ensemble whose decision rule consisted in evaluating the nearest neighbour of the target image and if the neighbour is close enough then the image is classified with the same label. On the other hand, if the target image is far from each one of the new low-represented faces, then the convolutional network is used for the classification. They partecipated to the MS-Celeb1M Low-shot Face Recognition Challenge and they managed to reach a 99.58% accuracy on the highly represented images and a 92.64% coverage at precision equal to 99% on the low represented data. They managed to win this challenge, leaving the second classified team to 77.48% on the low represented data, although with an accuracy of 99.80% on the highly represented data.

Choi et al. [101] proposed an ensemble of CNNs with different preprocessings for face recognition exploiting the fact that Gabor filters seem to extract relevant features from human faces. They applied diffrent Gabor filters to the same image and then used each image to train a different network. Then, they used the corresponding network at test time. However, their approach only reaches 93.6% accuracy.

3.3 Skin lesion recognition

Skin lesion segmentation is one of the most popular applications in computer vision. Esteva et al. [102] showed that a trained convolutional network could detect skin cancer with an accuracy similar to the one of a trained dermatologist.

Harangi et al. [103] proposed a simple ensemble of AlexNet [104], VGGNet [23], GoogLeNet [32] for skin lesion classification and showed that the ensemble of the three networks outperformed every single method. A wider version of the ensemble was later proposed, comprehending also ResNet-50 [26], and the performance futher improved.

Mahbod et al. [105] used three different CNN architectures, two different EfficientNets [106] and SeResNeXt-50 [107] trained on images cropped at different sizes to create an ensemble. They reached the second place on the ISIC 2018 competition (Task 3) [108] on skin lesion classification. (2020)

Ahmed et al. [109] also used an ensemble of different networks for the ISIC 2019 skin classification challenge (Task 3). In their case, they used very large networks like Xception [28], Inception-ResNet-V2 [38], and NasNetLarge [110]. Their results are comparable to the best submissions on the final leaderboard, although they did not partecipate to the competition.

Gessert et al. [111] also partecipated the ISIC 2019 challenge, however for tasks 1 and 2 that also involved the metadata of the patients in the classification. They used an ensemble of EfficientNets [106], since these networks are well-suited for problems were the input images have different resolutions. They also used a weighted loss to mitigate class imbalance. They managed to win both the challenges with their ensemble.

3.4 Computer-aided Diagnosis from X-Ray and MRI

Computer-aided diagnosis is one of the most relevant topics in computer vision applications. X-Ray and CT images are a powerful diagnosis tool for many pathologies. The main difference between X-Ray and MRI images is that X-Rays only have one channel, while MRI images have a large number of channels corresponding to different depths.

Hashmi et al. [112] created an ensemble of multiple pretrained networks for pneumonia detection using chest X-Ray scans. They used ResNet18 [26], Xception [28], InceptionV3 [27], DenseNet121 [2] and MobileNetV3 [24] pretrained on ImageNet and report an AUC score of 0.9976 on the Guangzhou Women and Children's Medical Center pneumonia dataset [113].

Rahman et al. [114] proposed a very similar approach but used a different set of networks for transfer learning. They used AlexNet, ResNet18, DenseNet201 and SqueezeNet. They reach a 0.98 AUC score, but on a different dataset [115] than before, so the comparison is not fair.

Chouhan et al. [116] also proposed a different ensemble using AlexNet, DenseNet121, InceptionV3, ResNet18 and GoogleNet. They reached an accuracy of 96.4% with a recall of 99.62% on [113].

Ferreira et al. [117] created a multi-view ensemble by training two different networks on chest X-Ray images for pneumonia detection. They preprocessed every image in two different ways and used the two images to train two different VGG16. The first preprocessing consists in cropping the image at the chest level, while the second one consists in enhancing the contrast of the image and then use a segmentator to extract the lungs. The ensemble of the two networks outperforms the stand-alone approaches.

Gooseen et al. [118] proposed an ensemble of convolutional neural networks, multiple instance learning, and fully convolutional networks for the detection of pneumothorax. They reached an AUC score of 0.96 on the large scale Chestx-ray8 dataset [119].

After the outbreak of Covid-19, a large number of papers tried to classify Covid-19 pneumonia using convolutional networks.

Ahmad et al. [120] proposed an ensemble of MobileNet [24], and InceptionV3 [27] for the classification of Covid-19. They reached an F-score of 96.49%.

Afifi et al. [121] proposed a novel attention mechanism that could be added to any CNN and applied it to DenseNet161, InceptionV4 and ResNet18. The models with the attention managed to outperform the baselines on Covid-19 classification. Besides, the ensemble of the the original and modifed models outperform both.

Deb et al. [122] also used an ensemble of pretrained CNNs to classify Covid-19. However, they trained NasNet, DenseNet121 and Mobilenet and concatenated the activations of their last layer to train a SVM. They report a classification accuracy of 91.99% on a dataset made of healthy images as well as normal and Covid-19 pneumonia.

Vantaggiato et al. [123] also proposed an ensemble of ResNeXt-50, Inception-v3, and DenseNet-161 to classify Covid-19 on two different datasets. They obtain very high results on the dataset they propose, reaching 100% accuracy in a three-label classification problem.

Speaking of CT images, Kang et al. [124] used 13 different networks pretrained on ImageNet for brain tumor classification. Among these networks, they selected the best three and made an ensemble by concatenating the activations of their last layers and feeding them into a classifier. They tested multiple classifiers and the best one turned out to be a SVM.

Islam et al. [125] proposed an ensemble of CNN models for Alzheimer's disease diagnosis. Their method is based on a multiple CNN architectures that they created for their paper. They report an improved level of classification in particular for the early stage of the disease.

Pan et al. [126] also created an ensemble of CNNs for early detection of Alzheimer's disease. They created a custom CNN architecture and they trained multiple networks using bagging. Besides, they used different slices of the data to train different CNNs, hence each model learns specific features of the data distribution. After that, the prediction of every model is averaged to return a probability distribution of the output class.

Leming et al. [127] proposed an ensemble method to classify function MRI connectivity matrices with the task of recognizing autism. Since the classification is hard and the dataset quite unbalanced, they used multi-task learning to also classify the gender of the subject and whehter he or she was resting or not. In this way, they achieved a better generalization and an AUC of 0.67 in autism recognition, which is a good result in this task.

Rasti et al. [128] used dynamic contrast-enhanced MRI to diagnose breast cancer with an ensemble of CNNs. They use four networks, three of which are trained on different regions of interest of an iage, while the fourth is trained to decide the weights to assign to each one of the former three CNNs. They report a performance improvement over a baseline ensemble that only averages the results of multiple CNNs.

3.5 Optical Charachter Recognition

Optical charachter recognition (OCR) consists in reading handwritten characters and words. This is a topic of great interest, since many documents, especially old ones, are handwritten. MNIST [1] classification is an early example of this problem, which is now particularly challenging for those languages whose alphabet is very large, like Chinese.

Lee et al. [129] proposed an ensemble of CNNs for handwritten music symbol recognition. They used a simple ensemble of three different CNNs trained on the whole training set.

Awni et al. [130] trained an ensemble of three independent ResNet50 architectures to for Arabic handwritten charachter recogntion, and showed that their method could outperform each of the single ResNets.

Roohi et al. [131] an ensemble of simple CNN architectures using bagging for the classification of Persian handwritten charachters. They also show that the ensemble method outperforms every single architecture.

Zhong et al. [132] created an ensemble of GoogleNet architectures trained for handwritten Chinese characther recognition. They also extracted Gabor filters from the images as a preprocessing before the training of CNNs and showed that this method managed to outperform single networks and that the feature extraction was an effective tool in this context.

Wu et al. [100] created and ensemble of CNNs for subtitle detection and recognition from East-Asian videos. They concatenated the output of 10 CNNs and fed it into a SVM for classification. They report a classification accuracy that was the state-of-the-art at the time they were writing.

3.6 Audio Classification

Audio files classification is an important application of CNNs. Despite audio signals are sequences that should not be fed into CNNs, they are tipically mapped into spectrograms before training the CNNs, hence they are mapped into an image.

Palanisamy et al. [133] created an ensemble of ImageNet pretrained DenseNets for audio classification on the ESC-50 dataset for enviromental sound classification [134]. They reached a 92.89% accuracy and showed that, although all the networks had the same weights at the beginning, the stochasticity of the training led them to have indepependent outputs and their ensemble outperformed the single networks.

Kahl et al. [135] created an ensemble of seven networks training different models with different architectures and different data augmentation policies. They submitted their results to the BirdCLEF 2017 competition [136] reaching a 0,605 mean average precision.

Nanni et al. [19] created an ensemble of multiple CNNs and multiple handcrafted features to classify audio signals on a variety of datasets. They showed that their "off-the-shelf" ensemble managed to outperform the state-of-the-art on different datasets without the need of any hyperparameter tuning.

Zimmerman et al. [137] created an ensemble of ten simple CNNs architectures for hate speech recogntion. They managed to reach an F-1 score of 0.7783 which was nearly 0.03 points higher than the state-of-the-art at the time.

Deng et al. [138] created an ensemble of networks for phone speech recognition by training multiple models and by learning posterior weights to average the results of the trained models. They experimented a linear and a log-linear model and showed that this last method was the most effective.

Lopez-Meyer et al. [139] proposed an ensemble of aclnet, aclresnet [140] and VGG12 for their submission to the Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge 2020 [141]. Interestingly, they report that ensembling the best stand-alone models does not return the best possibile ensemble.

Yang et al. [142] trained six different CNNs on raw spectrograms and on cropped spectrograms and used them for the 2019 DCASE challenge. They reached a 0.84 mean average precision on the submission.

Huang et al. [143] also created an ensemble of networks for DCASE 2019 using four networks pretrained on Audioset [144]. They reached a 81.3% classification accuracy and showed that the ensemble method outperformed the single networks on the validation set.

Sakashita et al. [145] proposed an ensemble of networks trained on different spectrograms preprocessed with different algorithms to train the same architectures multiple times. They report a mean average precision of 0.769 on the DCASE 2017 challenge.

Inoue et al. [146] created an ensemble of networks by using an approach similar to bagging. They trained four networks using each time a different train-validation split, and then they averaged the results of the networks for inference. They reached a 89.95% of macro-averaged F1-score on the Task 5 of the DCASE 2018 challenge.

Speaking of a different application of sound classification, Noman et al. [147] created an ensemble of 1D CNNs trained on raw audio signals and 2D CNNs trained on spectrograms to detect abnormal heart sounds. Their ensemble outperformed every single method and reached 89.22% accuracy and 89.94% sensitivity.

Humayun et al. [148] also created an ensemble for phonocardiogram sounds using 1D CNNs and a SVM on top of handcrafted features. They used pretrained CNNs for transfer learning and reached a 64% accuracy on the benchmark dataset.

Chapter 4

Data Augmentation for Building Ensembles

Data augmentation is one of the most popular technique to improve the performance of deep neural networks. Very large networks require a very large number of training samples to be able to generalize. Since the training samples are often not enough for that, researches started to use various data augmentation techniques to generate new training images starting from the original ones. The secret of data augmentation is to generate new samples exploiting properties of the inputs or invariances of the task, so that the new sample is plausible and its label can be easily found given the label of the original input. I shall now provide saome examples of data augmentation in the field of computer vision and sound analysis, which are the two topics investigated in this thesis.

In computer vision, the main transformations used for data augmentation exploit the geometrical property of the images: small image translations and rotation, random flips and random stretches of the original image all provide different samples that might be quite far in L^2-norm, but semantically very similar. Besides, color based transformations such as contrast enhancement or random changes in luminosity can all lead to new samples.

For audio-related tasks, the transformations must reflect the main properties of audio signals. For example, pitch shift only changes its frequency. Besides, one can use delays or cuts of the input. The speed of the audio sample can also be reduced or increased by a small percentage. In general, most tasks are independent from most of these transformations, that make them suitable for data augmentation.

Standard data augmentation is done by training a network on the original and all the transformed images until the network reaches convergence. However, this is not the only way. In [149*], I showed that data augmentation can be exploited to train an ensemble instead of training a single network. I trained different network using different data augmentation strategies and

Net	Data Aug	CH	HE	LY	AVG
ResNet50	App1	98.15	94.42	87.73	93.43
	App2	94.77	94.53	87.73	92.34
	App3	94.77	**95.70**	88.00	92.82
	App4	96.62	93.37	89.97	93.32
	App4-sa	95.08	93.26	86.40	91.58
	ENS	**98.77**	94.77	**92.00**	**95.18**
DenseNet	App1	**99.38**	96.00	88.00	94.59
	App2	98.77	95.20	91.47	95.14
	App3	98.46	95.30	90.13	94.63
	App4	98.46	95.93	89.07	94.48
	App4-sa	98.15	95.81	86.93	93.63
	ENS	**99.38**	**96.40**	**93.60**	**96.46**

Table 4.1: Performance of stand-alone CNN

showed that their ensemble managed to outperform the single networks.

In order to augment the total images in the datasets, I exploited different transofmations, divided in four protocols. The first (App1) applies a random reflection of the image in the left-right direction. The second (App2) also reflects the image in the top-bottom direction and linearly scales the image along both axes by two different numbers sampled between 1 and 2. The third protocol, which is named App4, also rotates the images by $r \in [-10, 10]$ degrees and translates them by $t \in [0, 5]$ pixels. The last protocol (App3) extends the third one by applying vertical and horizontal shear of $s_o, s_v \in [0, 30]$.

I test the ensemble on three competitive bioimage datasets, which are all publicly available:

- CH: the CHINESE HAMSTER OVARY CELLS [150] contains 327 fluorescent microscopy images of size, divided into 5 classes. Images are available at: http://ome.grc.nia.nih.gov/iicbu2008/hela/index.html.
- HE: the 2D HELA dataset [150] contains 862 images of HeLa cells acquired by fluorescence microscope and divided into 10 classes. Images are available at: http://ome.grc.nia.nih.gov/iicbu2008/hela/index.html.
- LY: Lymphoma [151] dataset of malignant lymphoma of three subtypes. Images are available at: https://ome.grc.nia.nih.gov/iicbu2008.

In Table 4, I compare the results of ResNet50 and DenseNet trained using different augmentation strategies. I also report the performance of:

- App4-sa, which is the single best CNN, obtained using different batch sizes and learning rates in a given dataset using App4 as the data augmentation method. This testing protocol is obviously biased, since

I select the best network on the test set, which might lead to overfit, however, I only use App4-sa as a baseline method to be compared with the ensemble.

- ENS: this is the fusion by sum rule among App1, App2, App3, and App4 and does not include App4-sa.

It is clear from Table 4 that the ensemble obtained using different data augmentation strategies outperforms the ensemble of networks trained using always the same policy. Besides, it also outperforms the biased protocol App4-sa.

Based on this result, I explored many ways to ensemble networks trained with different augmentation strategies. I shall now describe two applications in the field of image classification and of audio classification.

4.1 Ensemble of Saliency Methods for Image Classification

Insect pests cause a major part of crop damages world-wide. These damages could be prevented and lead to a higher efficiency of the agricolture. Pests might also damage the machinery and the equipment used for agricolture. Hence, solving this issue would lead to a more sustainable agricolture that needs less resources to produce the same food. A key step in this direction consists in being able to recognize dangerous pests before it is too late. However, this task requires a constant monitoring by someone able to distinguish hundreds of species of insects, which is unfeasible. Hence, recent years saw a growing interest in computer vision methods to recognize dangerous pests. I show some examples in Figure 4.1

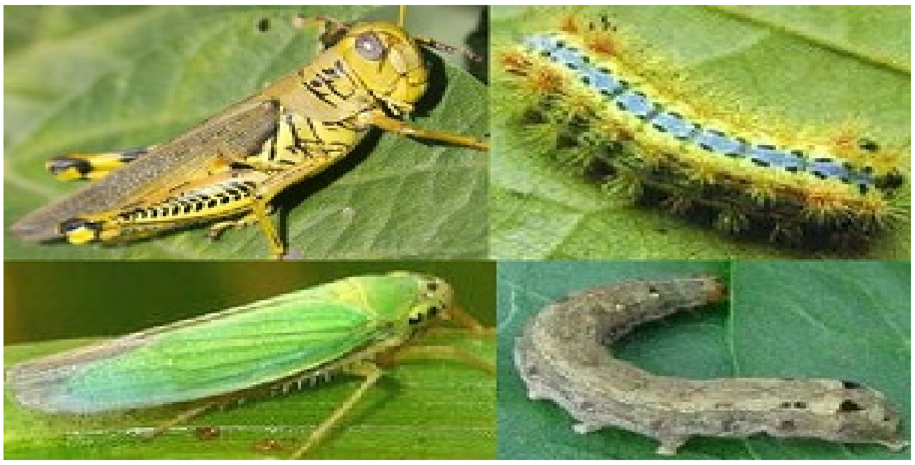


Figure 4.1: Examples of pest images

In [152*] I tested a new method to classify insect pests based on convolutional neural networks (CNNs) and saliency methods. Saliency methods are computer vision algorithms that give a score to each pixel depending on how relevant it is in the image. These algorithms are based on the fact that the human eye does not look at a whole image to understand its content, but it focuses on the most relevant parts [153]. Saliency methods often try to mimic this behaviour and can be used as a preprocessing step for computer vision algorithms. Using a saliency method as a preprocessing step can be seen as a data augmentation strategy, since it creates a new sample representing the same insect. However, training an architecture on the original and on the saliency augmented images might not be the best solution, since they clearly belong to different data manifolds. Hence, I used the saliency methods as a preprocessing step for both the training and the test images of a given network. Then, I created an ensemble using these networks. I used three different saliency methods that I shall now describe.

4.1.1 Saliency Methods

The first method is called GBVS, which stands for Graph-Based Visual Saliency. The idea to find the importance of the pixels is to define a transition probability $p\left((i,j);(k,l)\right)$ between every couple of pixels $\left((i,j);(k,l)\right)$ in the image. This probability is defined as

$$p\left((i,j);(k,l)\right) = d\left((i,j)||(k,l)\right) \cdot F(i-k, j-k) \tag{4.1}$$

where

$$d\left((i,j)||(k,l)\right) = \left|\log\left(\frac{I(i,j)}{I(k,l)}\right)\right|, \tag{4.2}$$

where $I(i,j)$ is the pixel intensity at (i,j). This function is minimized when the intensity of a couple of pixels is the same, hence it increases the transition probability among different pixels

$$F(a,b) = \exp\left(-\frac{a^2+b^2}{2\sigma^2}\right). \tag{4.3}$$

This function exponentially decreases the transition probability of far couple of pixels. The key step in GBVS is to use this transition probability to create a Markov chain on the pixels space and find their stationary distribution, which is unique under very easy assumptions on the input image. The probability of a pixel in the stationary distribution is high in those pixels where the transition probability from the other pixels is large. Hence, it is high if a pixel is different enough from other important and close pixels. An example of GBVS mask can be seen in Figure 4.2.

The second method described in the paper is Cluster-based Saliency Detection (COS). This method is based on a two layer clustering that is

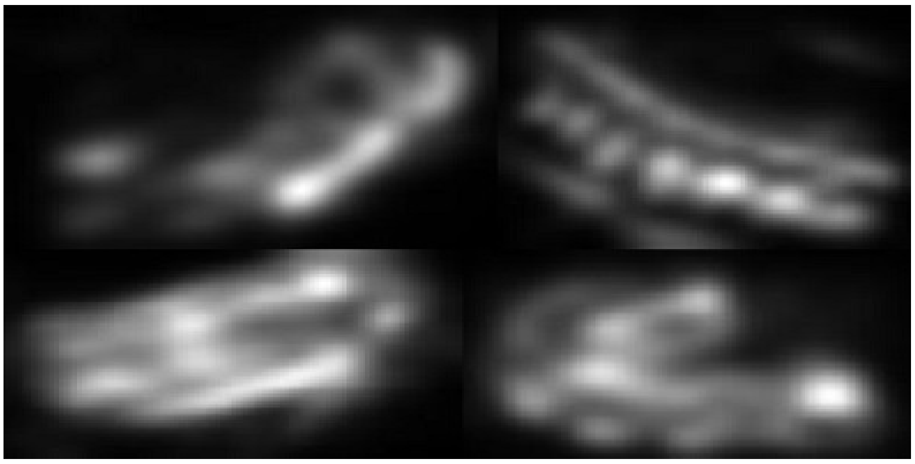


Figure 4.2: Examples of GBVS masks

done at image level at first, and at dataset level after that. The main idea is to extract three different image features at pixel level. The first one is based on the contrast of the image. The second one is based on the position of the pixel, since pixels in the center are usually more important than the ones on the sides. The third feature, which is also based on the first two, operates on multiple images and is designed to find correspondences among the features in different images. An example of COS mask can be seen in Figure 4.3.

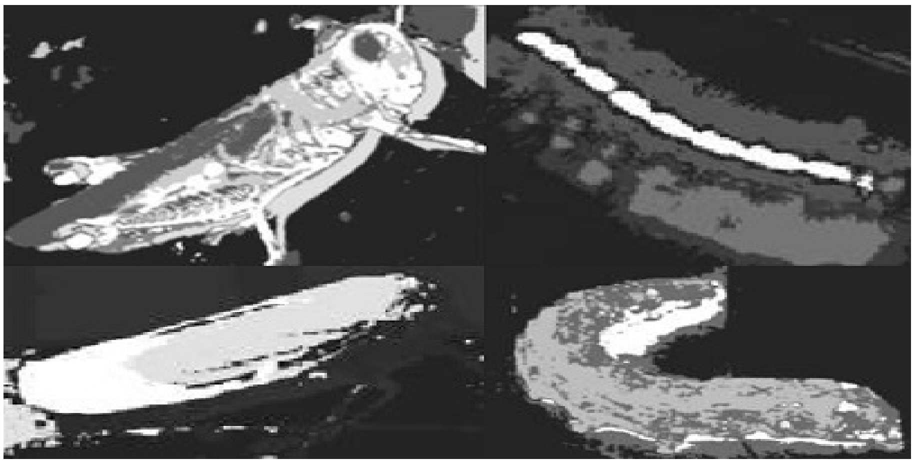


Figure 4.3: Examples of COS masks

The third saliency method is based on the outliers in the frequency domain and is called Spectral Residual (SPE). The Fourier transforms of the images are computed, then, based on the assumption that their spectrum should share similar trends, one can look for statistical singularities in the images and highlight those specific frequencies. Then, one can invert the Fourier transform on those frequencies and obtain a saliency measure on the pixels in the origial image space. An example of SPE mask can be seen in Figure 4.4.

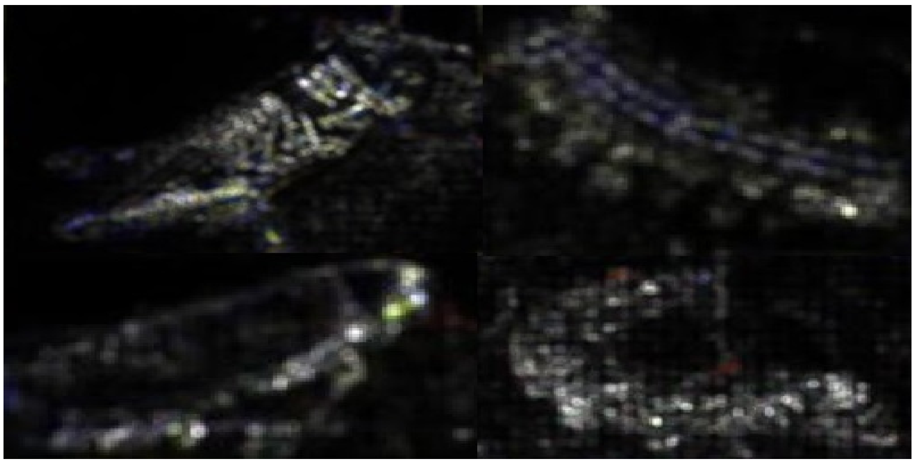


Figure 4.4: Examples of SPE masks

The output of each saliency method is called saliency map. For each saliency method, I create three new images, hence I create nine new images for each original one. Hence, I create nine different new image datasets, in addition to the original one. From every saliency map I find three new images: in the first case (FG), I turn to black all the pixels whose saliency is below a threshold; in the second case (ROI), I crop the image to highlight only the region where the saliency level are high enough; the third method (FG-ROI) is a combination of the first two.

4.1.2 Results

For each one the ten datasets, I train five different neural networks. AlexNet [104] and GoogleNet [32] are simple neural networks and very easy to train. They are among the first networks proposed after the outbreak of deep learning. MobileNetv2 [24] and ShuffleNet[154] are two very light neural networks suitable for mobile applications. They have performances that are similar to the ones of AlexNet, despite being of a much smaller size. The last network is DenseNet201 [2], which is a very large network whose performance is much better than the ones already mentioned. Its main feature is that every layer is connected to each other. A general representation of the method is in Figure 4.5

I perfomed experiments on two insect pest datasets. The first one is a small dataset proposed by Deng et al. [155] , which is made by 563 images divided in 10 classes, the most frequent having 72 samples and the less fre-

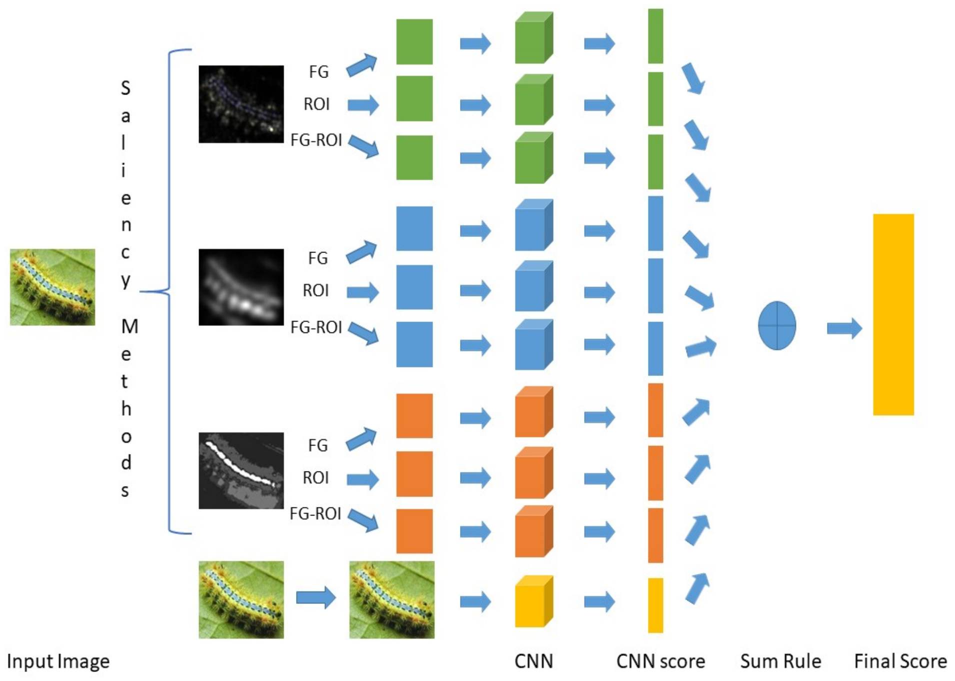


Figure 4.5: Representation of the proposed ensemble.

quent 40 samples. The other dataset is much larger and it is the competitive IP102 dataset [156], containing 5222 images divided in 102 different classes. The dataset is split in 45095 images for training, 7508 for validation and 22169 for testing. The least represented class in the dataset has 71 samples, while the most represented class contains 5740 images.

The results reported in Tables 4.2 and 4.3 show the accuracy of the single networks and the ensembles. For stand-alone networks, the row represents the preprocessing used, if any, and the column represents the architecture. "Fusion Sum" is the performance of the ensemble created using the same network trained on all the preprocessings. "All" is the ensemble created with all the networks. The term minus used in the name of the ensemble indicates that the networks after the minus sign are not included.

The ensembles without DenseNet are evaluated because they are much lighter than the others. In general, the performance show state-of-the-art results obtained by the best performing ensembles in both datasets.

4.2 Ensembles of Different Data Augmentations

Some of the most relevant applications of sound classification include speech recognition [158], music classification [159], biometric identification [160], and environmental sound recognition [134].

SMALL		Alex	Google	Mobile	Shuffle	Dense
COS	FG	78.67	79.78	76.19	77.68	85.03
	ROI	77.35	79.12	74.97	76.80	83.92
	FGROI	83.15	87.46	84.14	86.46	88.34
GBVS	FG	79.61	82.98	76.63	80.55	83.04
	ROI	77.57	81.16	76.80	78.56	77.62
	FGROI	81.16	84.14	81.44	83.59	86.74
SPE	FG	73.26	76.63	65.97	69.78	69.45
	ROI	73.31	74.42	64.97	69.34	69.06
	FGROI	86.69	87.73	83.31	84.81	88.51
Original Image		83.76	85.69	81.55	86.52	90.22
Fusion Sum		88.23	90.77	89.72	91.66	92.10
Fusion Sum minus FGROI		86.30	87.68	85.97	89.06	91.60
All minus Dense		91.88				
All minus FGROI and Dense		90.83				
All minus SPE and Dense		89.89				
All		**92.43**				
[155]		85.50				

Table 4.2: Performance of stand-alone CNN on the small dataset

SMALL		Alex	Google	Mobile	Shuffle	Dense
COS	FG	44.08	47.31	48.28	45.47	52.45
	ROI	41.58	44.52	46.77	42.99	51.98
	FGROI	47.01	49.08	51.51	49.50	56.14
GBVS	FG	42.86	48.13	49.92	46.35	54.56
	ROI	45.77	49.54	50.79	48.08	55.65
	FGROI	49.75	51.97	53.82	51.27	57.86
SPE	FG	35.90	41.68	43.68	39.22	47.77
	ROI	37.73	41.00	43.17	39.77	47.26
	FGROI	48.91	51.01	52.16	50.07	56.29
Original Image		51.79	53.80	55.35	52.45	58.76
Fusion Sum		54.60	56.13	58.92	56.43	**61.93**
Fusion Sum minus FGROI		53.63	55.49	57.94	55.63	61.21
All minus Dense		59.65				
All minus SPE and Dense		59.73				
All		61.44				
[156]		49.50				
[157]		55.24				

Table 4.3: Performance of stand-alone CNN on IP102

The latest advances in deep learning have been applied to all the fields of sound recognition. For example, monitoring the biodiversity of animals by recognizing the species from the sound they make has been improved by developments in animal sound recognition.

In this thesis, I focus on animal sound recognition and environmental sound recognition. These topics have already been explored in the literature, In [161], they try to asses the biodiversity by classifying marine animals and fishes by their sound. They use CNNs to classify short segments of sound after extracting frequency spectrograms.

Environment sound classification is one of the most competitive tasks in sound recognition. The Environmental Sound Classification (ESC-50) dataset [162] contains 2000 sound segments divided into 50 classes. In [163], a deep CNN obtained better results than humans on this data set. Other work of interest in this topic includes [164, 165, 166, 167, 168]. For a more comprehensive survey on sound classification up to the present day, I refer to [169].

In this section, I shall introduce the work presented in [170*] and show how to use data augmentation to train different CNN models for audio recognition. Audio signals can be augmented directly, or after they have already been transformed into spectrograms, and they can affect the time and the frequency of the audio. In [171], for example, they applied many different augmentation techniques for the BirdCLEF 2018 dataset classification. They cut the signals, changed their speed and shift their pitch. Besides, they added random noise to the sample. This augmentation led to a nearly 10% improvement in the accuracy. In [172], samples were also generated by summing different samples belonging to the same class. The idea behind this is that if two different segments contain multiple birds tweeting, their sum will also contain multiple birds tweeting, and it will be a very different sample from both the original ones. Here, I use Audiogmenter, which is a MATLAB library that I created [173*] to generate a very large number of new samples and to extract different kinds of spectrograms from the audio signals.

4.2.1 Audio Image Representations

I shall now describe the methods to extract the spectrograms and to perform data augmentation. I start from the spectrograms.

1. The Discrete Gabor Transform (DGT): it is a Short-Time Fourier Transform (STFT) with a Gaussian kernel as the window function. It can be defined for continuous time as an integral between the signal and a Gaussian in the complex space:

$$G(\tau, \omega) = \frac{1}{\sigma^2} \int_{-\infty}^{\infty} x(t) e^{-\pi\sigma^2(t-\tau)^2} e^{i\omega t} \tag{4.4}$$

where ω is a frequency, τ is a time, σ^2 is the variance of the Gaussian and $x(\cdot)$ is the signal. The discrete version of the DGT is a discrete approximation of this integral. I refer to [174] for further details on the implementation.

2. Mel spectrogram (MEL) [175]: it is evaluated by extracting the coefficients relative to the compositional frequencies with STFT. It is done using a filter-bank that keeps the frequencies that are the most relevant for the human ear and discards the others. All the filters in the filter-bank are triangular-shaped. Then, a conversion of the frequencies is done using the following formula:

$$m = 2595 \log_{10}(1 + 700f) \tag{4.5}$$

3. Gammatone (GA) band-pass filter: this is a bank of Gammatone filters whose bandwidth increases with the increasing central frequency. The idea is again to mimic the human ear and the fact that the most suitable metric for a distance between frequencies is logarithmic. The response can be defined as

$$h_i(t) = \begin{cases} at^{n-1}e^{-nB_it}\cos(2\pi\omega_i t + \phi) & t \geq 0 \\ 0 \quad t < 0 \end{cases} \tag{4.6}$$

where ω_i, ϕ_i are the central frequency and the phase of the filter. The constant a controls the gain, and n is the order of the filter. B_i controls the decay of the responce over the time.

4. Cochleagram (CO): this algorithm aims to highlight the frequencies heard by the human cochlea. At the beginning, one must filter the original signal with a gammatone filter bank and then the signal must be split into overlapping windows. For each window and every frequency, the energy of the signal is calculated and normalized.

Every spectrogam is then mapped into a gray-scale image and normalized so that its minimum and maximum are respectively 0 and 255.

Several data augmentation methods are exploited to obtain the best performance. Different augmentation protocols have been used. I shall now list them. Some of the samples are shown in Figures 4.6, 4.7

4.2.2 Standard Signal Data Augmentation (SGN)

In this augmentation protocol, each transformation happens with a 50% probability

1. Modify the speed of the signal by a factor in $[0.8, 1.2]$

2. Shift the pitch of the signal by a random number in $[-2, 2]$.
3. Modify the volume of the signal by a factor in $[-3, 3]$ dB.
4. Add random noise of volume sampled in $[0, 10]$ dB.
5. Anticipate or delay the start of the segment by a time sampled in $[-5, 5]$ ms.

4.2.3 Short Signal Augmentation (SSA)

In this protocol, all the augmentations are applied to the original signal indpendently. This means that I get 10 new images for each original image.

1. Apply wow resampling, which is a variant of pitch shift where the shift is not constant.
2. Add random noise of volume sampled in $[0, 10]$ dB.
3. Normalize the audio signal by leaving 10% of the samples out of [-1, 1], with the out-of-range samples clipped to the sign of x;
4. Modify the speed of the signal by at most 15%.
5. Apply quadratic distortion to the signal; The following distortion is applied five times:
$$s_o = \sin(2\pi s_i) \tag{4.7}$$
6. Increase the gain up to 10 dB.
7. Break the signal into two pieces and swap them.
8. Apply Dinamic Range Compression. It increases the lower intesities and decreases the higher intensities of the signal. I refer to [59] for further details.
9. Apply pitch shift by decrasing the frequencies of two semitones.

4.2.4 Super Signal Augmentation (SSiA)

In this protocol, twenty-nine images are created for each original one, while iterating multiple times the following pipeline of transformations:

1. Apply wow resampling
2. Modify the speed on the signal by a percentage in $[-5, 5]$.
3. Modify the gain by a number of dB between $[-0.5, 0.5]$.
4. Break the signal into two pieces and swap them.
5. Apply pitch shift by $[-0.5, 0.5]$

4.2.5 Time Scale Modification (TSM)

In this protocol I apply the five algorithms contained in the TSM toolbox [60].The objective of the methods in this tool box is to change the speed, while preserving the pitch. I shall now describe the algorithms.

1. OverLap Add (OLA): it covers the input signals with overlapping windows of size H_a and maps them into windows of size H_s. The number H_a depends on the implementation of the algorithm, while the ratio $\alpha = \frac{H_s}{H_a}$ is the speed-up factor, that I set to either 0.8 and 1.5. I copied the value from the original paper where the toolbox was presented;

2. Waveform Similarity OverLap Add (WSOLA): this is an upgrade of OLA since the window overlap has some tolerance to better model those signals where there is a difference of phase;

3. Phase Vocoder does the same thing, but it works on the frequency domain after a Fourier transform. This time, frequency windows and not time windows are used to change the tempo, the inverse of the Fourier transform is done.

4. Phase Vocoder with identity phase locking: it is a modification of Phase Vocoder, but the frequencies are not supposed to be independent of each other, as it happens in Phase Vocoder. I refer to for further details [62];

5. Harmonic-Percussive Source Separation (HPSS): it decomposes an audio signal into its harmonic and percussive sound components. They are then respectively processed by Phase Vocoder with identity locking and OLA, since they are the most suitable algorithms for harmonic and percussive sounds. Then they are summed to obtain the original sound with the new speed.

4.2.6 Short Spectrogram Augmentation (SSpA)

SSpA are augmentations applied directly to spectrograms. As in SSiA I create one new image for every augmentation protocol.

1. Apply time and pitch shift

2. Apply Vocal Track Length Normalization (VTLN) that divides a spectrogram into non-overlapping windows and normalizes the frequency in each window.

3. Cut the spectrograms into two windows ans swap them.

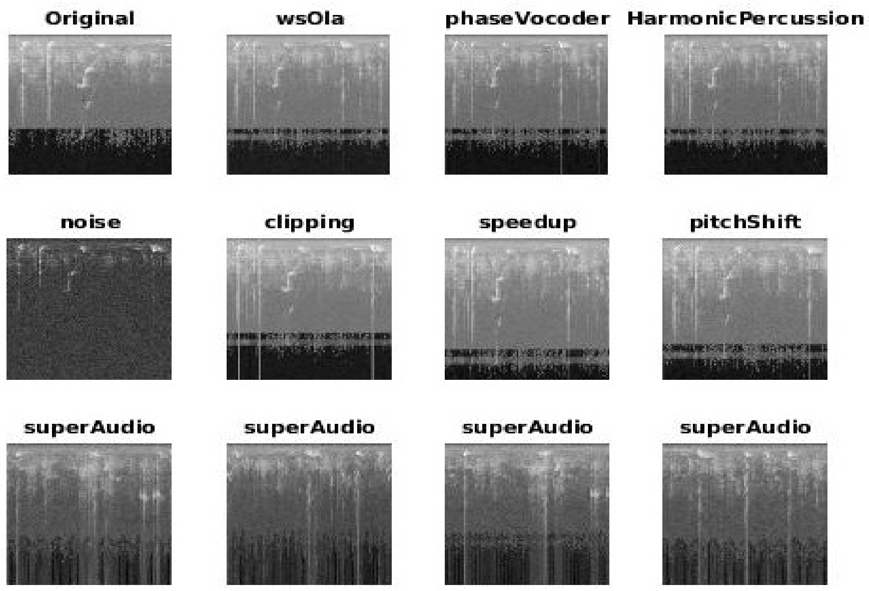


Figure 4.6: Examples of spectrogram augmentations

4. Apply Thin-Spline Image Warping (TPS-Warp) to a spectrogram. This algorithm consists in moving by a random number of pixels a subset of the points of an image and then change the other pixels to adjust the whole image to the random movement.

5. Apply pointwise random noise to the image, with probability equal to 0.3 on every point, In those pixels, the value of the intensity is multiplied by a random number extracted from a uniform distribution with average one and variance one.

4.2.7 Super Spectro Augmentation (SuSA)

In this protocol, I create twenty-nine new images from each original one by applying the following pipeline of augmentations, obtaining every time a different sample.

1. Apply random shift in the range $[-1, 1]$ semitones.

2. Apply VTLN.

3. Cut the window into two pieces and swap them as in the previous protocol.

4. cover with black pixels two columns and one row of the image.

5. Apply point-wise random noise as in the previous protocol.

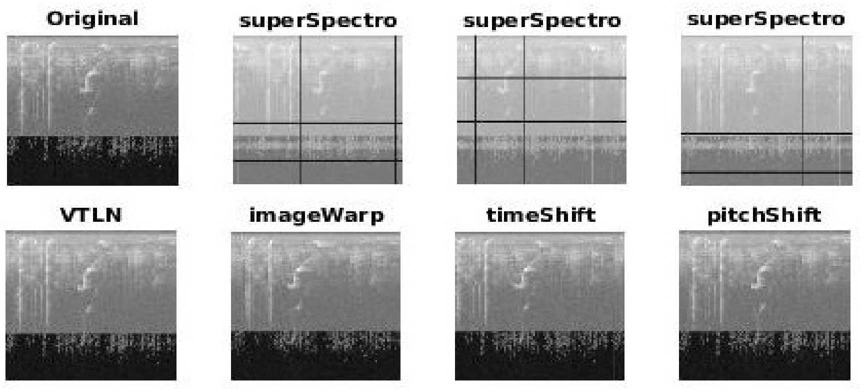


Figure 4.7: Examples of spectrogram augmentations

4.2.8 Results

Five different network architectures are trained on these datasets. These are AlexNet, GoogleNet, VGG16, VGG19, ResNet50, ResNet101 and Inception. All these network were pretrained on ImageNet, but I also used a GoogleNet trained on Places365 [176]. I also trained a VGG16 architecture with a batch size of 30. The rest of the networks were trained for 30 epochs, batch size of 60 and with a learning rate of 0.0001, except in the last layer where it is 0.001.

The results are evaluated on three dataset:

1. BIRDZ [47]: it is an audio dataset, containing 2762 recordings from the Xeno-canto Archive, 339 of which do not belong in any class.

2. CAT [41,46]: which is a dataset of 300 samples divided evenly in 10 classes dowloaded from Kaggle, YouTube and Flickr.

3. ESC-50 [4] is an environmental sound classification dataset containing 2000 samples evenly divided into 50 classes. They are roughly divided into Animals, Natural soundscapes and water sounds, Human non-speech, Interior and domestic sounds, exterior and urban noises.

In Tables 4.5, 4.4, 4.6, 4.7, I report the accuracy obtained using the data augmentation protocols described above and I compare it with the no augmentation baseline as well as with several state-of-the-art methods 4.8. Every network was trained for 30 epochs and using a leraning 0.0001 on every layer except the last fully connected layer that has a learning rate 10 times larger. The batch size is always equal to 30, except VGG16BS which is trained with a batch size of 60.

Six different ensembles are also reported in Tables 4.5, 4.4, 4.6 and 4.7.

CAT	NoAUG	SGN	SSA	SSpA	SSiA	SuSA	TSM
AlexNet	83.73	85.76	86.10	83.39	87.12	86.78	87.12
GoogleNet	82.98	86.10	87.80	83.39	86.78	85.08	87.80
VGG16	84.07	87.12	88.47	85.76	87.80	87.80	88.47
VGG19	83.05	85.42	87.80	84.75	86.10	86.10	89.15
ResNet50	79.32	81.36	85.42	76.95	85.08	82.03	87.12
ResNet101	80.34	84.75	85.42	75.59	82.03	73.56	86.78
Inception	79.66	82.71	—	66.44	—	84.07	86.10
GoogleNetPls	85.15	86.44	85.76	83.73	86.10	86.10	88.47
VGG16BS	—	86.10	88.14	86.78	89.49	86.10	89.15
FusionLocal	88.14	88.47	89.83	86.78	89.83	89.83	90.51
FusionShort	88.47						
FusionShortSuper	89.83						
FusionSuper	90.17						
FusionALL	89.83						
FusionSuperVGG16	89.83						

Table 4.4: Performance of the networks on the CAT dataset

The combination is done using the sum rule. I shall now list the abbreviations that I used in the tables:

1. FusionLocal: sum rule of CNNs where each one is trained with a different data augmentation method;
2. FusionShort: sum rule of all CNNs trained with SGN, SSA, and SSpA;
3. FusionShortSuper: sum rule of all CNNs trained with SGN, SSA, SSpA, SSiA, and SuSA;
4. FusionSuper: sum rule of all CNNs trained with SGN, SSiA, SuSA, and TSM;
5. FusionSuperVGG16: sum rule of VGG16 trained with SGN, SSiA, SuSA, and TSM;
6. FusionALL: sum rule of all CNNs trained with SGN, SSA, SSpA, SSiA, SuSA, and TSM.

Sometimes,VGG16 fails to converge; if it happened, I tried a second training. If it also failed, the results are not reported and are not used in the sum rule of the ensembles. In Tables 4.5, 4.4 and 4.6 the algorithms to map audio segments into images is a Discrete Gabor Transform.

In Table 4.8, the best ensembles are compared with the state-of-the-art. FusionGlobal is the sum of the CNNs of Fusion Super and the ones in Table 5. FusionGlobal-CO is FusionGlobal without the CNNs trained using CO to

BIRDZ	NoAUG	SGN	SSA	SSpA	SSiA	SuSA	TSM
AlexNet	94.48	94.96	95.40	94.02	95.05	95.76	88.51
GoogleNet	92.41	94.66	94.84	91.48	93.85	95.85	82.91
VGG16	95.30	95.59	95.60	94.69	95.44	96.18	94.63
VGG19	95.19	95.77	87.15	94.50	95.44	96.04	94.88
ResNet50	90.02	94.02	93.22	90.48	92.95	94.16	91.75
ResNet101	89.64	94.00	92.76	88.36	92.84	94.20	90.62
Inception	87.23	93.84	92.48	83.81	92.30	94.01	90.52
GoogleNetPls	92.94	94.81	95.10	92.43	94.76	95.80	86.91
VGG16BS	—	95.84	—-	94.91	95.81	96.31	94.78
FusionLocal	95.81	96.32	96.24	95.76	96.39	96.89	95.27
FusionShort	96.47						
FusionShortSuper	96.79						
FusionSuper	96.90						
FusionALL	96.89						
FusionSuperVGG16	96.78						

Table 4.5: Performance of the networks on the BIRD dataset

ESC-50	NoAUG	SGN	SSA	SSpA	SSiA	SuSA	TSM
AlexNet	60.80	72.75	73.85	65.75	73.30	64.65	70.95
GoogleNet	60.00	72.30	73.70	67.85	73.20	71.70	73.55
VGG16	71.60	79.40	80.90	75.95	79.35	77.85	79.05
VGG19	71.30	78.95	78.80	74.10	78.00	76.40	77.45
ResNet50	62.90	76.65	75.95	70.65	77.20	73.95	77.40
ResNet101	59.10	75.25	75.65	70.05	77.50	72.30	74.85
Inception	51.10	71.60	74.70	63.45	75.55	71.10	70.65
GoogleNetPls	63.60	75.15	76.10	71.35	74.00	71.60	73.55
VGG16BS	—	79.40	80.50	73.45	79.35	77.85	80.00
FusionLocal	75.95	84.75	85.30	80.25	85.25	82.25	85.30
FusionShort	86.45						
FusionShortSuper	87.15						
FusionSuper	87.55						
FusionALL	87.30						
FusionSuperVGG16	85.75						

Table 4.6: Performance of the networks on the ESC-50 dataset

	CAT			BIRD			ESC-50		
	GA	MEL	CO	GA	MEL	CO	GA	MEL	CO
AlexNet	82.03	83.73	79.32	91.85	91.43	87.54	73.95	73.50	65.50
GoogleNet	74.07	84.07	77.97	90.71	88.96	86.95	73.75	73.25	66.15
VGG16	83.39	86.10	80.00	92.65	93.17	88.82	77.60	79.20	66.75
VGG19	85.76	83.73	77.97	92.93	93.22	89.07	76.40	77.55	65.85
ResNet50	82.03	83.05	75.93	90.87	90.74	86.98	75.80	76.05	67.75
ResNet101	82.71	82.37	79.32	91.15	91.00	87.28	75.00	74.80	64.90
Inception	79.66	84.75	77.63	89.53	89.86	87.35	73.95	72.55	67.50
GoogleNetPls	83.05	82.71	77.63	90.88	88.31	86.75	73.60	75.50	68.70
VGG16BS	85.42	87.80	81.02	93.09	93.22	89.43	77.80	78.95	67.50
FusionLocal	87.46	88.47	82.37	93.76	93.97	90.57	81.90	83.80	73.25

Table 4.7: Performance of the ensembles with different preprocessings

Descriptor	BIRDZ	CAT	ESC-50
[177*]	96.45	89.15	85.85
FusionGlobal	96.82	90.51	88.65
FusionGlobal-CO	97.00	90.51	88.55
[178]	96.3	—	—
[159]	95.1	—	—
[179]	93.6	—	—
[180]	—	87.7	—
[181]	—	91.1	—
[181] - CNN	—	90.8	—
[164]	96.7	—	—
[182]	—	—	94.10
[183]	—	—	89.50
[168]	—	—	87.10
[167]	—	—	88.50
[184]	—	—	84.90
[163]	—	—	86.50
[166]	—	—	83.50
[185]	—	—	83.50
[165]	—	—	81.95
Human Accuracy [162]	—	—	81.30

Table 4.8: state-of-the-art comparisons

represent the signal. I can draw the following concolusions, based on these results:

1. There does not seem to be a clear winner among the data augmentation strategies. Methods that perform well on a dataset, do not necessarily transfer to other datasets.
2. VGG16 and VGG19 seem to be the best architectures.
3. DGT seems to be the best signal representation algorithm.
4. The fusion of different CNNs is better than the stand-alone approaches.
5. Although DGT outperformed the other image representations, adding the networks trained with the other images representations to the ensemble allows to improve the performance of the classifier.
6. The approach in [182] outperforms our results, although using additional data for training, hence the comparison is not fair.

Experimental results show that ensembles of fine-tuned CNNs trained with different data augmentation and signal representation methods allow to obtain results comparable with the state-of-the-art on multiple benchmarks, including on the ESC-50 dataset, without the need of any specific hyperparamenter tuning and are well suited to be an off-the-shelf method that can be applied to audio classification.

4.3 Discussion

In this chapter, I evaluated methods to create ensembles of neural networks using different techniques of data augmentation. In both cases, I compared ensembles of different sizes, which is not a totally fair comparison. However, it turns out that the best ensemble is not necessarily the largest, although it is usually among the best ensembles.

In the case of pest classification using saliency maps, the best ensemble is the largest one when it comes to the smallest dataset, suggesting that a small dataset might lead to overfitting of the single networks and that a large ensemble might reduce the noise in the classification. In the larger IP102 dataset, however, the best ensemble only contains the DenseNets, which are the largest architectures and best performing ones on IP102. This suggests, as it could be expected, that deleting the worst networks in an ensemble leads to an improvement in the overall performance and that a key for a good ensemble is having models whose stand-alone perfomance is similar.

Besides, one can see that the ensemble of multiple MobileNets outperforms the one of the stand-alone DenseNet in both datasets. Although this comparison might seem unfair, MobileNetv2 has a number of flops of approximately 0.3 billion, while Densenet201 has 8 billion flops. Hence, an ensemble of 10 Mobilenets has fewer flops than a single DenseNet201.

Similar conclusions can be drawn from the audio classification. The ensembles have different sizes, but the best performing ensemble among the ones that only use a single spectrogram does not contain all the networks, but only the best performing ones, which are the ones where a Super augmentation is applied. This is true for all the three datasets. However, the absolute best ensembles are created using different image representations of the audio signal. I hypothize that this is due to the fact that those representations highlight different features of the audio sample and let the networks learn different features, while preserving a good performance.

In general, in both cases creating an ensemble using different data augmentation techniques allowed me to reach the state-of-the-art in both pest datasets and on the BIRD dataset. Besides, I reached the second best result on the very competitive ESC-50 among the ones reported in the literature at that time, and I reached the third best resultthe CAT dataset. However, it worth noticing that the ensemble described here obtain very good results across multiple and diverse datasets.

Chapter 5

Ensembles of Different Activation Functions

Activation functions are one of the key features of modern neural networks. The research branch investigating which properties an activation shoud have to increase a network performance is a hot topic in deep learning and I shall summarize some of its main findings in this chapter.

The main topic of the chapter, however consists in investigating how different activation functions can be used to create an ensemble of neural networks. Using the taxonomy of Chapter 2, this technique falls among the ensembles created using different models. I shall start by introducing some of the most popular activation functions in the literature.

5.1 Activation Functions for Neural Networks

At first, neural networks were trained the hyperbolic tangent as activation function. However, it has the drawback that this function is bounded in both directions and rapidly saturates, leading to a very slow training. Hence Glorot et al. [186] showed that deep networks could be efficiently trained using the well known ReLU activation [187], although it is not differentiable everywhere. Using this activation, Krizhevsky et al. [104] managed to train the famous AlexNet architecture. At that point, ReLU was the most popular activation function in computer vision and research focused on finding variants that could improve its performance. The task is not easy and I could say that ReLU is still the most popular activation due to its fast computation. The most popular modification is leaky ReLU, which was proposed in [188]. It consists of a simple modification of ReLU which introduces a small, positive slope for negative numbers. Given $a > 0$ a small real number, leaky ReLU is defined as

$$f(x) = a\min(0, x) + \max(0, x). \tag{5.1}$$

The advantage of leaky ReLU is that it is more difficult for the optimization process to get stuck into flat regions of the loss landscape. In [189] this problem is also addressed, but in this case the slope of the activation decreases exponentially and saturates at a value $-a$ with $a > 0$. This new function is called Exponential Linear Unit (ELU) and the definition of the function is

$$f(x) = a \min(0, e^x - 1) + \max(0, x). \tag{5.2}$$

Klambauer et al. [190] proposed a very small modification of ELU consisting of a multiplication term and named it Scaled Exponential Linear Unit (SELU). It is defined as

$$f(x) = \lambda (a \min(0, e^x - 1) + \max(0, x)). \tag{5.3}$$

where a, λ are positive numbers. Although this modification seems nearly irrelevant, the idea behind it is that by scaling a, λ the output distribution of SELU can have zero mean and unit variance.

These activations have no learnable parameters. However, they can be easily introduced in the definition of an activation function, requiring more training resources, while leaving the complexity of the inference unchanged. He et al. [191] proposed parametric ReLU (PReLU), which is a modification of leaky ReLU where the parameter a is learnable and is different in every channel. This allows the network to have a larger representation power and was one of the keys to reach super-human performance on ImageNet. Its authos report that PReLU always outperforms non-learnable activations on the training set, but sometimes fail to generalize on the test set.

Many other learnable activations have been proposed. S-shaped ReLU (SReLU) was proposed in [192] and, as its name suggest, is a learnable modification of ReLU whose shape recall the letter S. Its formal definition depends on the four learnable variables a_l, a_r, t_l, t_r, with $t_l < 0 < t_r$ and obeys the following formula

$$f(x) = \begin{cases} t_l + a_l(x - t_l) & x \leq t_l \\ x & t_l \leq x \leq t_r \\ t_r + a_r(x - t_r) & t_r \leq x. \end{cases} \tag{5.4}$$

Agostinelli et al. [193] proposed a Adaptive Piecewise Linear Unit (APLU), which is a learnable function that contains as many learnable parameters as one wants and, using the right amount of parameters, it can represent any piecewise linear function. This function is a piecewise linear function whose slopes and non-differentiability points are learnable. It is defined as follows:

$$f(x) = \max(0, x) + \sum_{i=1}^{n} a_i max(0, -x + b_i) \tag{5.5}$$

where a_i and b_i are learnable parameters. Their creators suggest to train this function with a small $\mathbb{L}^2-$penalty on the slopes in order to avoid divergence.

Ramachandran et al. [194] proposed the Swish activation function, which is defined as

$$f(x) = x \cdot \sigma(\beta x) \tag{5.6}$$

where σ is the sigmoid function and β is a parameter that can or cannot be learnable. This function was found by its creators using reinforcement learning and looking for the best performing function made by different combinations of different building blocks.

Parametric Deformable Exponential Linear Unit (PDELU) was introduced by Cheng et al. [195] and is defined as:

$$f(x) = \begin{cases} x & if x \geq 0 \\ \alpha \cdot \left([1 + (1-t)x]^{\frac{1}{1-t}} - 1 \right) & if x < 0 \end{cases} \tag{5.7}$$

It is designed to have zero mean, and this should improve the training speed.

Mish is an activation function introduced in [196]. It is defined as

$$f(x) = x \tanh \left(\log \left(1 + e^{-\alpha x} \right) \right) \tag{5.8}$$

where α is a learnable parameter.

Soft Root Sign (SRS) was introduced in [197]. It is defined as:

$$f(x) = \frac{x}{\frac{x}{\alpha} + e^{\frac{-x}{\beta}}} \tag{5.9}$$

where α and β are learnable and non-negative parameters. As PDELU, this function is also designed to have zero mean for fast training. A modification of SRS is the so-called Soft Learnable which is defined as:

$$\begin{cases} x & x \geq 0 \\ \alpha \log \left(\frac{1+e^{\beta x}}{2} \right) & x < 0 \end{cases} \tag{5.10}$$

where α and β are positive parameters. The parameter β is designed to be optionally learnable, and I shall use both versions in this thesis, the former without (SoftLearnable) and the latter with β learnable (SoftLearnable2).

It was not feasible for us to search the best hyperparameters for every activation function, also considering that I do not use validation sets, hence I use the paramenters that are reported to be the best in every paper where these activations are introduced.

j	1	2	3	4	5	6	7
b_j	512	256	768	128	384	640	896
λ_j	512	256	256	128	128	128	128

Table 5.1: Fixed parameters of MeLU with $k = 8$ and $M = 256$

5.1.1 Mexican and Gaussian Linear Units

Mexican ReLU (MeLU) [198*], is a variant of ReLU that comes from the Mexican hat functions. Mexican hat functions are defined as:

$$h_{b,\lambda}(x) = \max\left(0, \lambda - |x - b|\right) \tag{5.11}$$

where b, λ are real numbers. MeLU is defined as:

$$f(x) = \max(0, x) + a_0 \min(0, x) + \sum_{i=1}^{k-1} a_i h_{b_i,\lambda_i} \tag{5.12}$$

which is the weighted sum of PReLU and Mexican hat functions. The parameters k, a_i, λ_i are fixed. Besides, the parameters a_i, λ_i are set according to a bisection algorithm. All the parameters a_i are learnable.

The name Mexican hat derives from its shape, since it is null when $|x - b| > a$ and has a triangular shape with the maximum in a. Its drawing recalls a Mexican hat.

The first hyperparameter is M, which is used to find the parameters b_i, λ_i. The first Mexican hat function reaches the maximum in $2 \cdot M$ and becomes zero for $x < 0$ and $x > 4 \cdot M$. The next two functions becomes zero on the intervals $[0, 2 \cdot M]$ and $[2 \cdot M, 4 \cdot M]$. The idea is that the next functions every times split in half by the previous ones, in order to get a better and better approximation. The representation power of MeLU is very similar to the one of APLU, and in this sense these functions can be considered similar, although the non-differentiability points of MeLU are not learnable. In Table 5.1 with the first parameters of MeLU are reported.

In the rest of the thesis MeLU will be used with two different values of k, which are 4 and 8. If $k = 8$ I refer to this function are wider MeLU (wMeLU).

The main propoerties of MeLU are the following

1. MeLU is continuous and piecewise differentiable.

2. If all the learnable parameters are zero, MeLU is equal to ReLU. This is a useful property because MeLU can replace ReLU in a pretrained network allowing an efficient transfer learning.

3. The gradient of MeLU is just the sum of the gradients of its component functions.

4. The gradient of MeLU with respect to the learnable parameters is bounded and it is null outside a small region, except for the first parameter, which is the one relative to $\min(0, x)$.

Gaussian Linear Unit, also called GaLU [199*], is an activation based on Gaussian-like functions and inspired by MeLU:

$$\begin{aligned}\phi_g^{b,\lambda} &= \max(\lambda \cdot M - |x - b \cdot M|, 0) \\ &\quad + \min(x - a \cdot M - 2\lambda \cdot M - \lambda \cdot M, 0)\end{aligned} \tag{5.13}$$

where λ, b, M are real numbers. GaLU is defined as:

$$f(x) = max(0, x) + a_0 \min(0, x) \sum_{i=1}^{k-1} a_i \phi_{b_i, \lambda_i} \tag{5.14}$$

which is the equivalent of MeLU for Gaussian-type functions and their parameters are the same as ReLU. In the experiments I call smaller GaLU (sGaLU) the implementation with $k = 2$ and GaLU the one with $k = 4$.

5.2 Changing the Activation Functions

In this section I present a simple way to create ensembles of different and well-performing networks. The method was introduced by Maguolo et al. in [198*]. The idea consists in choosing a network architecture and selecting a pool of different activation functions. Then, for every function in the pool, I can build a new network whose activation function is the new one in every activation layer. In this way, multiple different networks have been created from just on original architecture. It is worth noticing that if the original network is pretrained on a large dataset, all the new networks can exploit transfer learning. In particular, this holds for those learnable activation that have a possibile initialization that coincides with ReLU, i.e: for most of the learnable activations introduced before. I call this approach Fusion.

A different approach that I introduced [199*] consists in a very similar idea, that has the advantage to create a potentially much larger ensemble. This technique consists in substituting all the activations in the pool with a new one randomly selected from the pool in every layer. The big difference is that this time the activation is different in every layer. Since this procedure is random, it potentially yields $n_p^{N_l}$ different networks, where N_l, n_p are respectively the number of activation layers and the number of activations in the pool. Since they both are easily larger than 10, there are billions of different architectures that can be found. In general, I create ensembles of size smaller than 100.

Dataset	Description	Classes	Samples
CH	Chinese Hamster Ovary Cells [150]	5	327
HE	2D HELA dataset [150]	10	862
LO	Locate Endogenous dataset [200].	10	502
TR	Locate Transfected dataset [200].	11	553
RN	Fly Cell dataset [151]	10	200
TB	Aging of Terminal bulb [151]: images of C. elegans terminal bulb at 7 ages	7	970
LY	Lymphoma dataset [151]	3	375
MA	Muscle aging [151]. C. elegans muscles at 4 ages	4	237
LG	Liver gender [151]. Liver tissue sections from 6-month male and female mice on a caloric restriction diet. The 2 classes are male and female.	2	265
LA	Liver aging [151]. This dataset shows liver tissue sections from female mice on ad-libitum diet of 4 ages	4	529
CO	histological images of human colorectal cancer [201].	8	5000
BGR	breast grading carcinoma [202].	3	300
LAR	Laryngeal dataset [200]	3	1320

Table 5.2: Datasets used in the experiments

5.3 Results

In this section I report the experiments that I ran in [198*] and in [199*]. I tested the stand-alone networks as well as various ensembles. Besides, MeLU and GaLU are activation functions introduced in those papers for the first time. In Table 5.3, I introduce the datasets used in the experiments.

In Table 5.3, I report the results of the stand alone activation functions. It is clear that there is not a clear winner, although wMeLU is the best on average when M is set to 255, which is the maximum value of the input and the choice suggested by its authors. It performs worse when M is set to 1. The experiments were run using ResNet50. With terms StocSmall and StocLarge I mean the networks coming from one iteration of the Stochastic algorithm of Section 5.2. The difference between the Small and Large depends on the pool of activations. The large pool contains all the activations described above, while small only contains ReLU, leaky ReLU, ELU, PReLU, APLU, MeLU (both values of k), SReLU, GaLU (both values of k).

In Table 5.3 I report the results of various ensembles. The nomenclature used in the table an be summarized as follows.

Activation	CH	HE	LO	TR	RN	TB	LY	MA	LG	LA	CO	BG	LAR	Avg	Rank
ReLU	93.5	89.9	95.6	90.0	55.0	58.5	77.9	90.0	93.0	85.1	94.9	88.7	87.1	84.55	6
leakyReLU	89.2	87.1	92.8	84.2	34.0	57.1	70.9	79.2	93.7	82.5	95.7	90.3	87.3	80.30	22
ELU	90.2	86.7	94.0	85.8	48.0	60.8	65.3	85.0	96.0	90.1	95.1	89.3	89.9	82.80	20
SReLU	91.4	85.6	92.6	83.3	30.0	55.9	69.3	75.0	88.0	82.1	95.7	89.0	89.5	79.02	24
APLU	92.3	87.1	93.2	80.9	25.0	54.1	67.2	76.7	93.0	82.7	95.5	90.3	88.9	78.99	25
GaLU	92.9	88.4	92.2	90.4	41.5	57.8	73.6	89.2	92.7	88.8	94.9	90.3	90.0	83.28	17
sGaLU	92.3	87.9	93.2	91.1	52.0	60.0	72.5	90.0	95.3	87.4	95.4	87.7	88.8	84.13	8
PReLU	92.0	85.4	91.4	81.6	33.5	57.1	68.8	76.3	88.3	82.1	95.7	88.7	89.6	79.26	23
MeLU	91.1	85.4	92.8	84.9	27.5	55.4	68.5	77.1	90.0	79.4	95.3	89.3	87.2	78.76	27
wMeLU	92.9	86.4	91.8	82.9	25.5	56.3	67.5	76.3	91.0	82.5	94.8	89.7	88.8	78.95	26
softLearnable2	93.9	87.3	93.6	92.5	46.0	60.3	69.0	89.5	94.6	86.1	95.0	89.6	87.0	83.41	15
softLearnable	94.1	87.4	93.4	90.3	47.0	59.1	67.7	88.3	95.0	85.5	95.5	89.3	88.2	83.13	19
pdeluLayer	94.1	87.2	92.0	91.6	51.5	56.7	70.9	89.5	96.3	86.6	95.0	89.6	88.1	83.77	12
learnableMishLayer	95.0	87.5	93.2	91.8	45.0	58.4	69.0	86.6	95.3	86.6	95.4	90.0	88.4	83.24	18
SRSLayer	93.2	88.8	93.4	91.0	51.5	60.1	69.8	88.7	95.0	86.4	95.7	88.3	89.4	83.94	10
swishLearnable	93.5	87.9	94.4	91.6	48.0	59.2	69.3	88.7	95.3	83.2	96.1	90.0	89.3	83.57	14
swishLayer	94.1	88.0	94.2	90.7	48.5	59.9	70.1	89.1	92.6	86.1	95.6	87.6	87.6	83.39	16
SReLU(255)	92.3	89.4	93.0	90.7	56.5	59.7	73.3	91.7	98.3	89.0	95.5	89.7	87.9	85.15	4
APLU(255)	95.1	89.2	93.6	90.7	47.5	56.9	75.2	89.2	97.3	87.1	95.7	89.7	89.5	84.35	7
GaLU(255)	92.9	87.2	92.0	91.3	47.5	60.1	74.1	87.9	96.0	86.9	95.6	89.3	87.7	83.73	13
sGaLU(255)	93.5	87.8	95.6	89.8	55.0	63.1	76.0	90.4	95.0	85.3	95.1	89.7	89.8	85.09	5
MeLU(255)	92.9	90.2	95.0	91.8	57.0	59.8	78.4	87.5	97.3	85.1	95.7	89.3	88.3	85.26	2
wMeLU(255)	94.5	89.3	94.2	92.2	54.0	61.9	75.7	89.2	97.0	88.6	95.6	87.7	88.7	85.27	1
StocSmall	90.2	90.0	94.2	91.6	54.5	62.0	77.3	90.8	95.7	90.5	95.1	89.0	87.1	85.23	3
StocSmall(255)	93.2	88.5	94.4	91.6	51.5	59.1	73.9	88.3	94.0	89.1	95.1	86.7	88.0	84.11	9
StocLarge(255)	94.1	87.2	93.0	87.3	54.5	60.1	72.3	89.2	94.7	83.6	94.6	89.0	89.9	83.80	11

Table 5.3: Performance of stand-alone methods

1. The terms that start with "Fus" indicate an ensemble performed using the Fusion method introduced in Section 5.2. The approaches start with "Sto" indicate that the fusion is done using the Stochastic method introduced in the same section-

2. The approaches devide among those that use a small pool of activations (Small) and a large pool of activations (Large). Besides, as a baseline, I also use ensemble made by ReLU networks only.

3. The number between parenthesis indicates the value of M for the networks in the ensemble, if it exists.

4. the plus operator indicates sum rule between the two methods.

5.4 Discussion

I now draw some conclusions based on the results repoted in Tables 5.3 and 5.4.

1. The ensemble models consistently outperform any stand-alone network. This was actually expected, but it is worth noticing.

2. ReLU is one of the best functions among the ones tested, but not the best. The one with the highest performance is wMeLU(255), that I proposed in [198*]. Also MeLU(255) performs very well. No other parameters other than $k = 4$ and $k = 8$ have been tested.

Activation	CH	HE	LO	TR	RN	TB	LY	MA	LG	LA	CO	BG	LAR	Avg	Rank
FusOldAS10	93.5	90.7	97.2	92.7	56.0	63.9	77.6	90.8	96.3	91.4	96.4	90.0	90.0	86.67	20
FusOldAS10(255)	95.1	91.3	96.2	94.2	63.0	64.9	78.7	92.5	97.7	87.6	96.5	89.7	89.8	87.46	13
FusFullAS16(255)	97.2	91.3	97.4	95.5	60.0	64.5	76.0	94.2	98.3	89.1	96.8	90.0	90.3	87.74	12
StoOldAS10	95.4	91.3	95.8	95.1	63.0	64.2	78.9	93.8	98.7	91.1	96.5	90.3	90.2	88.02	10
StoOldAS(255)	96.6	90.8	97.0	96.0	55.5	65.1	78.1	92.1	98.3	90.1	96.3	88.7	90.0	87.27	14
StoOldAS10(255)	96.9	91.2	96.8	96.2	58.5	66.6	79.7	92.5	98.3	91.6	96.6	89.7	91.1	88.13	9
StoOldAS15(255)	97.8	91.5	96.6	95.8	60.0	65.8	80.0	92.9	99.0	91.2	96.6	90.7	91.0	88.37	8
StoFullAS5(255)	98.1	92.3	96.6	95.5	64.0	64.6	83.2	93.8	99.0	92.6	96.6	91.3	92.1	89.20	6
StoFullAS10(255)	98.8	92.9	97.6	95.8	66.5	65.7	84.3	93.7	99.3	94.1	96.8	90.3	92.3	89.85	4
StoFullAS15(255)	98.8	93.4	97.8	96.4	65.5	66.9	85.6	92.9	99.7	94.3	96.6	91.3	92.3	90.11	2
StoBaseAS5(255)	99.4	91.0	97.6	95.6	61.5	64.1	80.0	88.3	94.7	91.8	96.5	90.3	90.4	87.78	11
StoBaseAS10(255)	99.4	93.5	97.8	95.6	65.5	65.8	81.3	89.6	96.3	94.9	96.7	91.0	90.8	89.09	7
StoBaseAS15(255)	99.4	93.9	98.0	96.0	64.5	66.4	83.2	90.0	96.0	93.9	96.7	92.0	91.3	89.33	5
StoBaseAS8(255)+ StoFullAS7(255)	99.4	93.8	98.0	96.0	67.5	66.3	83.5	92.1	98.0	95.1	96.7	91.7	91.7	89.98	3
StoBaseAS15(255)+ StoFullAS15(255)	99.4	93.8	98.0	96.5	67.5	67.0	85.9	91.2	98.7	94.9	96.9	92.0	92.3	90.31	1
FusOldAS3(255)	93.9	91.5	94.8	93.1	58.5	63.5	77.6	91.3	98.3	88.0	96.3	89.0	89.4	86.55	21
StoOldAS3(255)	96.3	90.9	95.6	95.1	54.0	62.9	78.7	92.5	98.7	90.9	96.2	90.0	90.5	87.10	16
FusRelu5	95.0	90.5	96.2	94.7	56.0	63.7	77.1	94.1	95.6	89.1	96.4	89.0	89.5	86.68	19
FusRelu10	94.5	91.6	95.8	94.5	56.5	64.5	76.0	93.3	97.7	89.1	96.6	89.6	90.2	86.91	17
FusRelu15	95.4	91.1	96.2	95.1	58.5	64.8	76.0	92.9	97.3	89.3	96.3	90.0	90.4	87.17	15
FusRelu30	95.4	91.5	96.2	94.7	59.0	63.9	75.7	92.5	97.3	88.2	96.4	89.0	90.1	86.91	18

Table 5.4: Performance of ensemble methods

3. The stochastic networks with multiple activations have a performance which is comparable to the original network.

4. Nevertheless, the Stochastic ensembles outperform the Fusion ensembles for ensembles of the same cardinality.

5. Both Fusion and Stochastic ensembles consistently outperform the ReLU ensembles with the same cardinality. This result in particular is important, because the baseline is strong, and it proves that introducing differences in the ensemble components allows to obtain high performances when the base networks have similar performances.

6. The ensemble size turns out to be a relevant factor, as larger ensembles tend to perform better. However, there seems to be a plateau around 15, as ensembles of that size perform as well as 30 networks ensembles.

This method to create ensembles is very general and can be applied to every neural network to boost its performance. The result proposed in this chapter does not rely on new theoretical analysis that I made, but I was able to prove its efficiency by experimenting on a large number of small datasets. This simple idea allows to have multiple versions of the same architecture to work well on small datasets, in particular when combined in an ensemble. This is particurarly useful with small datasets where neural networks often tend to overfit the data.

This does not seem to happen with ensemble models, despite the fact that I did not use any regularization methods and I did not have a validation set. The networks were trained for a fixed number of epochs on the new dataset.

Besides, I experimented with a large number of ensembles and compared them with other baseline ensembles, hence this is one of the few works in the literature that scientifically test the performance gain from introducing diversity in a class of deep neural networks ensembles.

Chapter 6

Ensembles of Different Trainings

Minimizing the training loss of CNNs is a task that is usually performed using gradient descent methods. The main example of this is stochastic gradient descent (SGD) [203]. However, SGD surely ends in global minima only if the optimization landscape is convex. However, in general, it is not and it has a large number of local minima where SGD might get stucked. This is why many modifications of SGD have been proposed [204, 205, 206, 207, 208, 209]. The goals of these modifications are finding better minima of the training loss and to converge to minima that generalize well in the test set. The real goal would be the second one, but it is also the most difficult. Hence, as it is common in deep learning, one can try to improve the performance on the test set by reaching the lowest possibile training loss, among those that can be reached with a given model. However, this does not always happens, and minimization algorithms that find better minima from the training loss point of view, might fail to generalize on the test set.

A straightforward modification of the usual SGD consists using its momentum to try to overcome local minima [204]. Including the momentum, and in general the past values of the gradient, is the idea of most SGD modifications, and of all the ones that I shall use here. For example, AdaGrad [206] and its modification Adadelta [207] also use the momentum of the gradient. The main idea behind these algorithms is decreasing the learning rate of variables with large partial derivatives.

Adam [208] is one of the most popular algorithms for training neural networks. The idea behind it is to keep track of the moving average of the gradient and of its square. If the gradient changes often, in particular if it changes its sign often, the average of the square of the gradient will be large even if the average of the gradient is low. Hence, dividing the gradient by the square root of the average of its squares allows to normalize the gradient and limits the training step to the learning rate. Adam reaches excellent results in minimizing the training loss, but it is reported to fail to generalize well on the test set with respect to a standard SGD [210]. In [211], they first train the networks on Adam to reach a good training loss, and then they switch to SGD to obtain better generalization. The modifications of Adam that have been proposed usually try to improve its generalization on the test set [209, 212, 213]. With Nadam [212], they introduced Nesterov momentum into Adam. AMSGrad [213] is designed to guarantee that the step size of the optimization path never increases. Recently, diffGrad [209] was introduced with the idea of making the step proportional to the gradient of the loss; in the experiments that [209] reported showed, diffGrad managed to reach state of the art results.

In this chapter, I show how the optimization process affects the networks performance: Following my work [214*], I experiment which optimization method performs better on three medical datasets on three image classification tasks, and I see the affect of creating an ensemble using the networks trained in different ways.

I take a given pretrained model, ResNet50, and I trained it several times to obtain a large ensemble of networks. As a preliminary section, I introduce the optimization algorithms that will be used.

6.1 Optimization Algorithms

6.1.1 Adam

Adam was introduced in [208]. The main idea behind it is to use the first two moments of the gradient to keep track of its mean and variance, and decrease the learning rate when the variance is large. Its update rule is a function of the value of the exponential moving averages of the gradient and its square. To be more precise, they define the moving averages m_t (the first moment) and u_t (the second moment) as:

$$m_t = \rho_1 m_{t-1} + (1 - \rho_1)\, g_t \tag{6.1}$$

$$u_t = \rho_2 u_{t-1} + (1 - \rho_2)\, g_t^2 \tag{6.2}$$

where g_t is the gradient at time t, the square on g_t is meant to be component-wise, ρ_1 and ρ_2 are hyperparameters that are usually set to 0.9 and 0.999 respectively, and control the forget rate of moving averages. The moments are initialized to 0 in the first step. i.e: $m_0 = u_0 = 0$. Since in the first steps the value of moving averages will be very small due to their initialization to zero, their value is then normalized:

$$\hat{m}_t = \frac{m_t}{(1 - \rho_1^t)} \tag{6.3}$$

$$\hat{u}_t = \frac{u_t}{(1-\rho_2^t)} \tag{6.4}$$

The final update for each parameter of the network is:

$$\theta_t = \theta_{t-1} - \lambda \frac{\hat{m}_t}{\sqrt{\hat{u}_t} + \epsilon} \tag{6.5}$$

where λ is the learning rate, ϵ is a very small positive number to avoid potential divisions by zero (usually set to 10^{-8}) and all the operations are component-wise.

While g_t might have positive or negative components, g_t^2 is always positive. Hence, if the gradient changes sign often, $\hat{m}_t$ will be much lower than $\sqrt{\hat{u}_t}$, lowering the resulting training step.

6.1.2 AMSGrad

AMSGrad is a modification of Adam proposed by Reddi et al. [213]. Its formulation remembers the maximum of the past squared gradients to update the parameters rather than exponential average as in Adam. Starting from Equations 6.1 and 6.2, AMSGrad is defined by the non decreasing parameter

$$\dot{u}_t = \max\left(\dot{u}_{t-1}, u_t\right) \tag{6.6}$$

The parameter update of the network is:

$$\theta_t = \theta_{t-1} - \lambda \frac{\hat{m}_t}{\sqrt{\dot{u}_t} + \epsilon} \tag{6.7}$$

6.1.3 diffGrad

The diffGrad algorithm was introduced by Dubey et al. [209]. It considers the difference of the gradient to decide the learning rate. The key steps of the algorithm are defined this way by the authors: they define the absolute difference of two consecutive steps of the gradient as:

$$\Delta g_t = |g_{t-1} - g_t| \tag{6.8}$$

The update of every parameter of the network θ_t the same as in Equation 6.5 where $\hat{m}_t, \hat{u}_t$ are those in Equations 6.3 and 6.4, and the learning rate is passed through the Sigmoid of Δg_t :

$$\xi_t = \sigma\left(\Delta g_t\right) \tag{6.9}$$

$$\theta_{t+1} = \theta_t - \lambda \xi_t \frac{\hat{m}_t}{\sqrt{\hat{u}_t} + \epsilon} \tag{6.10}$$

where $\sigma(\cdot)$ is the sigmoid function.

6.2 Methods

I shall compare several optimization methods for training ResNet50. Besides, they introduce and evaluate the following variants of the Adam optimization method:

- DGrad, which is a variant of diffGrad based on moving average of the element-wise squares of the gradients;
- Cos#1 and Cos#2, which are variants of DGrad based on a cyclic learning rate;

The proposed approaches have different methods for defining Δg_t, then 6.10 is applied as in diffGrad.

6.2.1 DGrad

DGrad uses the ideas of diffGrad defining the absolute difference between two steps of the gradient. The idea is that if the gradient rapidly changes, the training might be unstable and the step size should be reduced. It is defined as:

$$\Delta ag_t = |g_t - avg_t| \tag{6.11}$$

where avg_t is the moving average of the squares of the gradients. Then, they normalize Δag_t by its maximum as

$$\Delta a\hat{g}_t = \frac{\Delta ag_t}{\max\left(\Delta g_t\right)} \tag{6.12}$$

and ξ_t is defined as:

$$\xi_t = \sigma\left(4\Delta a\hat{g}_t\right) \tag{6.13}$$

The parameter update is as in Equation 6.5. The reason behind the 4x is to enlarge the range of the sigmoid function.

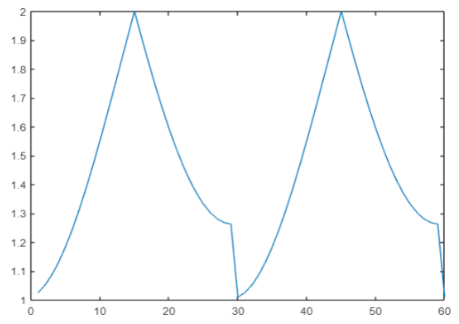
6.2.2 Cos#1 and Cos#2

These two algorithms are variants of Dgrad that are based on cyclic learning rates. The cycle of the learning rate is defined by the following equation:

$$lr_t = 2 - \left|\cos\left(\frac{\pi t}{steps}\right)\right| e^{-0.01(\mathrm{mod}(t,steps)+1)} \tag{6.14}$$

Where $steps = 30$ is the period. The cycle is shown in Figure, 6.2.2

Figure 6.1: Cyclic learning rate



In Cos#1 lr_i is a multiplier of $\Delta\hat{ag}_t$ in the definition of ξ_t, which now reads as:

$$\xi_t = \sigma\left(4 \cdot lr_t \cdot \Delta\hat{ag}_t\right) \tag{6.15}$$

In Cos#2 the definition of ξ_t also has an additive factor lra_t used to avoid a null learning rate:

$$lra_t = \left(\left|\cos\frac{\pi t}{steps}\right| e^{-0.01(\text{mod}(t,steps)+1)}\right) \tag{6.16}$$

$$\xi_t = \sigma\left(2lr_t\hat{ag}_t\right) + \sigma\left(4lra_t\right) - 0.5 \tag{6.17}$$

The update rule of each parameter θ_t of the network is performed as in Equation 6.2.2.

These are the hyperparamenters used to train all the CNNs:

- Batch size = 30;
- Number of epochs = 20;
- global learning rate = 0.001;
- gradient decay factor = 0.9;
- squared gradient decay factor = 0.999;
- loss function = cross entropy.

A simple data augmentation consisting of random reflections and random scales on both the axis are performed.

6.3 Experiments and Discussion

Here I list the datasets that I used to validate our approaches:

- HeLa, the 2D HELA dataset [215] contains 832 images belonging 10 classes. As in the existing literure, I use a 5 fold cross validation algorithm.
- BG, the Breast Grading Carcinoma [202] contains 300 color images divided into 3 labels that represent grades 1-3 of invasive ductal carcinoma of the breast. Again, I use a 5 fold cross validation protocol as in the existing literature.
- LAR, the Laryngeal data set [200] includes 1320 images of healthy and early-stage cancerous laryngeal tissues, split into 4 labels: He (healthy), Hbv (hypertrophic vessels), Le (leukoplakia) and IPCL (intrapapillary capillary loops). This dataset has already been divided into 3 folds by its authors.

In Table 6.1 I show the performance of the average on 7 experiments of the different approaches on the 3 datasets using a stand-alone ResNet50.

Table 6.1: Performance of stand-alone CNN

Accuracy	HeLA		BG		LAR	
	avg	std	avg	std	avg	std
SGD	92.09	0.66	88.33	1.19	93.03	1.11
Adam	55.90	29.66	86.57	5.66	92.15	5.39
diffGrad	79.00	19.83	89.00	4.77	93.01	3.05
DGrad	75.25	22.56	89.29	4.21	91.07	3.79
Cos#1	78.92	18.28	88.38	3.96	92.19	3.02
Cos#2	66.25	28.74	88.05	6.36	93.04	2.99

The performance reported in Table 6.1 show that Adam generalizes worse than SGD. On average, the variants of Adam obtain better accuracy than original Adam.

In Table 6.2 I report the performance of deep ensembles obtained by the fusion with the sum rule of several CNNs. The number of classifiers included in the ensembles is enclosed in parentheses.

From Table 6.2 I can see the following facts:

- The accuracy obtained by Adam-based approaches strongly improves using ensemble of CNNs
- The accuracy of DGrad(14) is very close to the one of SGD(14);
- Adam variants are better than Adam;

Table 6.2: Performance of ensemble of CNN

Accuracy	HeLA	BG	LAR
Adam(7)	74.30	89.67	96.29
diffGrad(7)	94.88	91.67	95.91
DGrad(7)	95.35	92.67	94.85
Cos#1(7)	95.00	92.67	95.38
Cos#2(7)	91.05	92.00	95.98
DGrad(7) + Cos#1(7)	95.81	92.33	95.91
DGrad(7) + Cos#1(7) + Cos#2(7)	96.05	92.67	96.06
DGrad(7) + Cos#1(7) + diffGrad(7)	96.28	92.33	96.06
DGrad(14)	95.70	92.67	95.68
DGrad(14) + Cos#1(7)	95.58	92.67	96.29
SGD(14)	96.05	94.00	94.70
SGD (7)	95.70	94.00	94.32
SGD(7) + DGrad(7)	96.16	94.00	95.38
SGD(14) + DGrad(7) + Cos#1(7)	96.74	94.33	95.98
SGD(14) + DGrad(7) + Cos#1(7) + diffGrad(7)	96.98	94.33	96.14

- The fusion of CNNs trained using different optimization algorithms (i.e. SGD with Adam) allows to improve the performance: SGD(7) + DGrad(7) is better than SGD(14) despite they have the same size. I believe that combining networks trained using different optimization methods is an effective method to create ensembles.

- SGD(14) + DGrad(7) + Cos#1(7) obtains an F-measure of 95.99 in LAR, while SGD(14) + DGrad(7) + Cos#1(7) + diffGrad(7) reaches an F-measure of 96.15. These results outperform the previous state-of-the-art that I reported in [37*] (95.20).

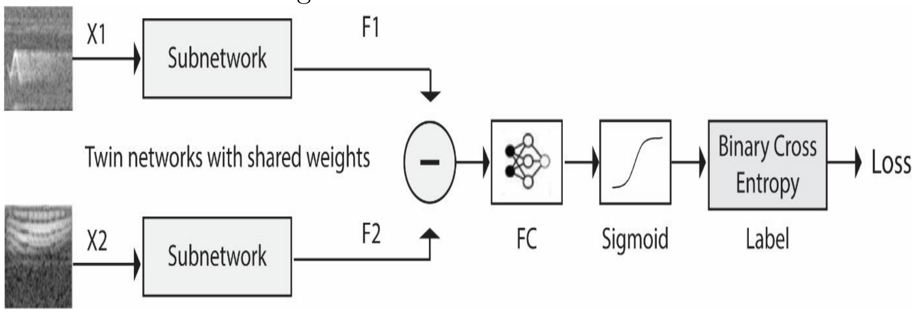
These results, however, are not as strong as the ones of the previous chapters in proving the efficiency of deep ensembles. My experients involved a smaller number of datasets and my aim was not to compare my work against other ensemble methods. This work focused on creating high performing ensembles for medical imaging applications. This goal was reached since these ensembles obtain results that were, in the worst case scenario, competitive with the state-of-the-art on the datasets that we considered. Hence, these methods have been proved to be very effective.

Chapter 7

Ensembles of Siamese Networks

Siamese Neural Networks are a class of neural networks used for classification [216]. They are made by two identical backbone networks that share their weights and whose input is a couple of input samples. In our case, they will be two identical CNNs and the inputs will be two images. Each backbone CNN processes one image, returning a vector of features for each input image. These two feature vectors are then subtracted and the absolute value is fed to a fully connected layer followed by a sigmoid function, returning a value in $(0, 1)$. The goal of the architecture is returning a value of 0 if the two images belong to the same class, and a value of 1 if their class is different. A scheme of a general Siamese network can be seen in Figure 7.1.

Figure 7.1: Siamese Network



In this chapter, I follow the work done in [217*] to create an ensemble of Siamese networks to build a dissimilarity space for image classification.

7.1 The Dissimilarity Space

In this Section I introduce the details of the method of the classification in all its steps. I first give a short overview of the method. A set of Siamese

Networks are trained on a given training set. I then iterate on the labels of the classification problem and I apply a clustering algorithm to all the training images. In this way, I find a subset of centroids, that I shall call prototypes, that I use to represent the training set. Then, for each image in the training set, I calculate the score of all our Siamese networks against all the prototypes of all the classes. In this way, for each input image I get a vector of dissimilarities that I use as a feature vector. I say that this vector belongs to a dissimilarity space. Then, I train a SVM to classify the sample. For inference, I perform the same dissimilarity evaluation and use the SVM.

The idea behind the prototype selection is that calculating the dissimilarity between every couple of images in the training set would require too much computation time, hence I need to select a representative subset of the training samples. This decision is unsupervised in the sense that it uses clustering algorithms like k-means and k-medoids [218] on the raw input, but among the inputs belonging to the same class, hence the training set split is supervised. The resulting prototypes can be evaluated against a new sample to test which ones are the most similar. Letting $x_1, ..., x_{k \cdot n}$ be the prototypes, where n is the number of classes and k is the number of prototypes selected for each class, the dissimilarity vector of an input image x is given by

$$F(x) = \begin{pmatrix} S(x, x_1) \\ \cdots \\ S(x, x_{kn}) \end{pmatrix} \tag{7.1}$$

where $S(\cdot, \cdot)$ is the Siamese network. On top on that, I train a SVM for classification. The scheme of method can be seen in Figure 7.2.

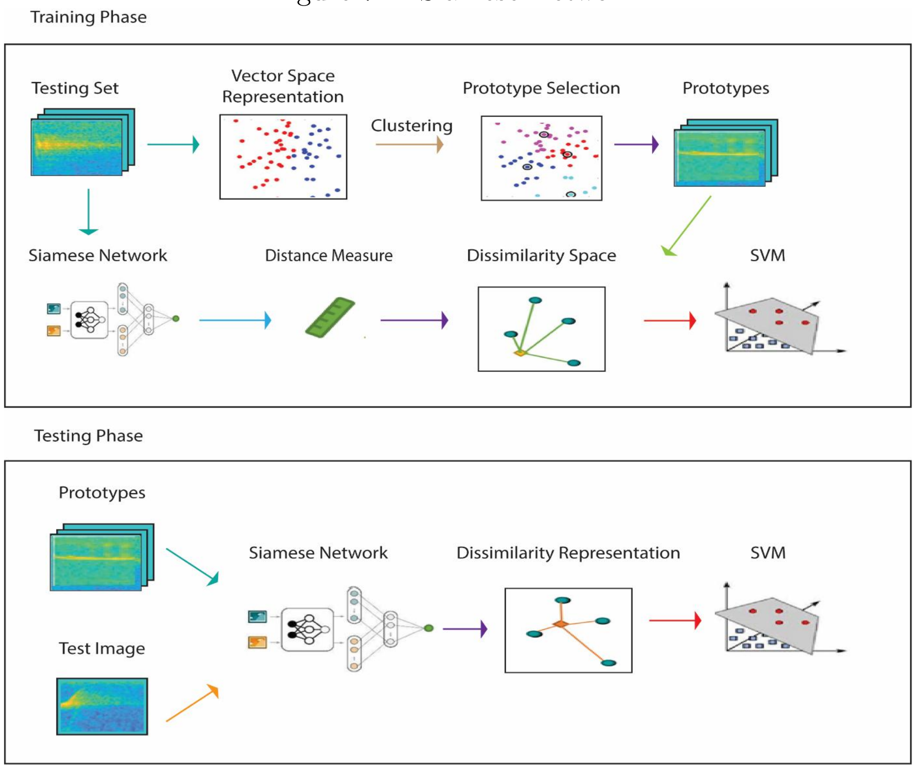
I apply this method to sound classification. In order to have input images suitable for our Siamese Networks, I extract the spectrograms of the audio samples. Besides, I also try to apply Heterogeneous Auto-Similarities of Characteristics (HASC) to the spectrograms to highlight different features of the input.

HASC [219] combines linear relations by covariances (COV) and nonlinear relations by entropy with mutual information (EMI). The reasons to use covariance matrices are the following: they have low dimension; they are robust to noise, except when it comes to outlier pixels; the covariance among two features can capture the features of the joint probability distribution. HASC tackles these issues by combining COV and EMI. I refer to [219] for a more detailed introduction to HASC.

7.2 Siamese Architectures

In order to create an ensemble, I used several Siamese architectures that I obtained by varying the backbone network. In this section I give the details of the networks that I used.

Figure 7.2: Siamese Network



The first one, summarized in Table 7.1 is a simple convolutional network that I created.

Layers	Filter Size	Number of Filters
2D Convolution	10×10	64
ReLU		
Max Pooling	2×2	
2D Convolution	7×7	128
ReLU		
Max Pooling	2×2	
2D Convolution	4×4	128
ReLU		
Max Pooling	2×2	
2D Convolution	5×5	64
ReLU		
Fully Connected	Returning a 4096-Dimensional Vector	

Table 7.1: First backbone

The second backbone network, that is summarized in Table 7.2, is deeper than the first one and is charachterized by the presence of leaky ReLU as activation function.

In Table 7.3, I report the layer structure of the third network. This is the shallowest architecture among the ones that I tested. The hidden layers dimension rapidly decreases thanks to many max pooling layers.

In Table 7.4 I describe the last network. In this backbone network I tried to artificially create a bottleneck in the number of channels. This is usually not recommended when creating CNN architectures, but my aim is to create four networks that are different enough from each other to be used in an ensemble. Hence, I inserted this new feature in the fourth network.

7.3 Clustering Methods

I use clustering of the images to find a subset of data samples that is represetative of the data set. I suppose that centroids contain the relevant features of patterns belonging to a cluster. The more centroids I use, the more information I keep about the data set, but doing so I also increase the computational requirements and the risk of overfitting.

I try both supervised and unsupervised approaches. We both use supervised and unsupervised versions of the algorithms, in the sense that all these algorithms are by their nature unsupervised, but I apply them to the training set as a whole and also to the different subsets of samples that have the same label. I shall now describe them.

Layers	Filter Size	Number of Filters
2D Convolution	5 × 5	64
leaky ReLU		
2D Convolution	5 × 5	64
leaky ReLU		
Max Pooling	2 × 2	
2D Convolution	3 × 3	128
leaky ReLU		
2D Convolution	3 × 3	128
laeky ReLU		
Max Pooling	2 × 2	
2D Convolution	4 × 4	128
leaky ReLU		
Max Pooling	2 × 2	
2D Convolution	5 × 5	64
leaky ReLU		
Fully Connected	Returning a 2048-Dimensional Vector	

Table 7.2: Second backbone

Layers	Filter Size	Number of Filters
2D Convolution	7 × 7	128
ReLU		
Max Pooling	2 × 2	
2D Convolution	5 × 5	128
ReLU		
Max Pooling	2 × 2	
Max Pooling	2 × 2	
Fully Connected	Returning a 4096-Dimensional Vector	

Table 7.3: Third backbone

Layers	Filter Size	Number of Filters
2D Convolution	7×7	128
ReLU		
Max Pooling	2×2	
Max Pooling	2×2	
2D Convolution	5×5	256
ReLU		
2D Convolution	3×3	64
Max Pooling	2×2	
2D Convolution	3×3	128
ReLU		
2D Convolution	5×5	64
ReLU		
Fully Connected	Returning a 4096-Dimensional Vector	

Table 7.4: Fourth backbone

7.3.1 K-Means

K-means is probably the most popular clustering algorithm. It divides the data points into k disjoint clusters based on which centroid is te closest to the a data point. I use the Euclidean distance as a metric.

The algorithm can be summarized as follows.

1. Randomly select a subset of data points to initialize the centroids.
2. Compute the distances between every data point and the centroids, then for each data point select the closest centroid and put it that centroid cluster.
3. Find the new centroid of every cluster based on which point is, on average the closest to other points of the cluster.
4. Repeat steps 2 and 3 until a maximum number of steps is reached or the configuration is stable.

7.3.2 K-Medoids

K-medoids is an algorithm very similar to k-means but different in the sense that it divides the data points into clusters by minimizing the sum of the distances between a given data point and the center of the cluster that contains that data. The main difference between k-means and k-mediods is that in k-mediods the centroid of the cluster is a point belonging to the cluster, while in k-means it can be a generic point of the Euclidean space. I shall now summarize how k-medoids work.

1. Randomly select the medoids among the data points.
2. Test each point as potential medoid of its cluster and select the one with the lowest average distance.
3. Assign every data point to the cluster represented by its closest medoid.
4. Repeat steps 2 and 3 until convergence or until a maximum number of steps is reached.

7.3.3 Spectral Clustering

This method consists in dividing the data into clusters using the adjacency matrix representing the undirected similarity graph of the data points. Each data point is a node of the graph and a conncection between two points means that the similarity among the two is above a threshold, which I set to 0. In order to describe the algorithm I first need to introduce some objects. I call M the similarity matrix that contains the similarity among the couples of data points. The degree matrix D, which is a diagonal matrix whose terms on the diagonal is the sum of the corresponding row in M. Lastly, I define the Laplacian matrix as $D - M$. I now summarize how the method works.

1. I start with a similarity definition given by nearest-neighbour, i.e: a point is similar to one of its nearest neighbours, if there is more than one.
2. Calculate the Laplacian matrix.
3. Let V be the matrix of the eigenvectors of L.
4. Apply k-means to the columns of V.
5. Cluster the original pattern according to the assignments of their corresponding rows.

7.3.4 Hierarchical Clustering

Hierarchical clustering divides the data point using a tree of clusters. The idea is to subsenquently divide the data points that are in a cluster into small subclusters. The algorithm can be summarized as follows:

1. Use a Euclidean dstance to find the similarity between every pair of data points in the dataset.
2. Divide the data points into two clusters using k-means.
3. Repeat steps 1 and 2 on the two clusters until a convergence criterion is satisfied or a given depth of the tree is reached.

7.4 Results

For the experimental evaluation, I used the BIRD and CAT datasets of Chapter 4. Here I report the experimental results obtained by several ensembles of Siamese networks trained with different preprocessings, with different clustering algorithms and with different network backbones.

In Table 7.5 I use K-means clustering for all the experiments. All the ensembles are fused together using the sum rule between methods trained with a different number of prototypes, but using only the first two network backbones. The data preprocessing also changes, since both classical spectrograms and HASC images are used in this context.

Name	Input image	Network topology	Clustering method	Clustering type	Prototypes	Classifiers	CAT	BIRD
Sup-1	Sp	NN1	K-means	S	15, 30, 45, 60	4	78.64	92.46
Sup-2	Sp	NN2	K-means	S	15, 30, 45, 60	4	76.95	92.74
UnS-1	Sp	NN1	K-means	U	15, 30, 45, 60	4	81.69	92.73
UnS-2	Sp	NN2	K-means	U	15, 30, 45, 60	4	75.25	92.80
HSup-1	HASC	NN1	K-means	S	15, 30, 45, 60	4	78.64	94.52
HSup-2	HASC	NN2	K-means	S	15, 30, 45, 60	4	81.69	93.22
HUnS-1	HASC	NN1	K-means	U	15, 30, 45, 60	4	79.32	94.53
HUnS-2	HASC	NN2	K-means	U	15, 30, 45, 60	4	81.36	92.97
FSp-1	Sp	NN1	K-means	S,U	15, 30, 45, 60	8	81.02	92.79
FSp-2	Sp	NN2	K-means	S,U	15, 30, 45, 60	8	76.95	92.77
FA-1	Sp,HASC	NN1	K-means	S,U	15, 30, 45, 60	16	82.37	94.50
FA-2	Sp,HASC	NN2	K-means	S,U	15, 30, 45, 60	16	83.73	94.11
FA1-2	Sp,HASC	NN1+NN2	K-means	S,U	15, 30, 45, 60	32	84.41	94.37

Table 7.5: Prototypes Ensembles

In Table 7.4 I compare the clustering algorithms. In this case, I use the second backbone network, which is the deepest one, to have the best results.

Name	Input image	Network topology	Clustering method	Clustering type	Prototypes	Classifiers	CAT	BIRD
	HASC	NN2	K-means	S	15, 30, 45, 60	4	81.69	93.22
	HASC	NN2	K-Med	S	15, 30, 45, 60	4	81.02	92.85
	HASC	NN2	Hier	S	15, 30, 45, 60	4	81.69	93.01
	HASC	NN2	Spect	S	15, 30, 45, 60	4	80.00	93.13
F-Clu	HASC	NN2	All	S	15, 30, 45, 60	16	82.03	93.37

Table 7.6: Comparing Clusters

In Table 7.4, I compare and fuse the network backbones using the supervised version of k-means clustering and using HASC images.

If I combine all the score obtained by the SVMs that I tested so far, I can obtain even better results: the accuracy on CAT becomes 85.76 and the one on bird becomes 95.08.

I now want to test how varying some of the parameters used so far I can create an effective ensemble. Since comparing ensembles and single SVMs might be unfair, I use a a baseline the ensemble obtained by using different network backbones.

Name	Input image	Network topology	Clustering method	Clustering type	Prototypes	Classifiers	CAT	BIRD
	HASC	NN1	K-means	S	15, 30, 45, 60	4	78.64	94.52
	HASC	NN2	K-means	S	15, 30, 45, 60	4	81.69	93.22
	HASC	NN3	K-means	S	15, 30, 45, 60	4	78.64	94.91
	HASC	NN4	K-means	S	15, 30, 45, 60	4	82.37	93.33
F-NN	HASC	All	K-means	S	15, 30, 45, 60	16	84.07	94.99

Table 7.7: Comparing Backbones

Name	Input image	Network topology	Clustering method	Clustering type	Prototypes	Classifiers	CAT	BIRD
HSup-1(1)	HASC	NN1	K-means	S	15	1	75.93	93.92
HSup-1(4)	HASC	NN1	K-means	S	15	1×4	81.69	94.50
HSup-1	HASC	NN1	K-means	S	15, 30, 45, 60	4×1	78.64	94.52
HSup-1(8)	HASC	NN1	K-means	S	15, 30, 45, 60	4×2	80.68	94.56
HSup-1(16)	HASC	NN1	K-means	S	15, 30, 45, 60	4×4	81.02	94.63
F-NN(4)	HASC	All	K-means	S	15	4	83.39	94.73
F-NN(8)	HASC	All	K-means	S	15, 30	8	84.07	94.90
F-NN	HASC	All	K-means	S	15, 30, 45, 60	16	84.07	94.99

Table 7.8: Comparing Ensembles

From Table 7.4 I see that varying the network topology was very useful to improve the ensemble performance: the comparison among ensembles of cardinality 4, 8 and 16, the one obtained from the topology NN1, and the ensembles obtained varying the topology of the Siamese Network show a clear boost in the accuracy of the latter (with the same cardinality). It is worth noticing the similar results of rows 2 and 3: these ensembles have a cardinality of four, but the first one is obtained by retraining the same model over and over, while the second has different numbers of prototypes: hence, that does not seem to be an important parameter to vary.

Method	CAT	BIRD
OLD [217*]	82.41	92.97
F-NN	84.07	94.99
GoogleNet	82.98	92.41
VGG16	84.07	95.30
VGG19	83.05	95.19
GoogleNetP365	85.15	92.94
eCNN	87.36	95.81
OLD+eCNN	87.76	95.95
F-NN+eCNN	88.47	96.03

Table 7.9: state-of-the-art results

I can now draw the following conclusions based on our experiments and on Table 7.4. The entry named eCNN is an ensemble of CNNs introduced in Chapter 4.

1. The best results is obtained using different network topologies.

2. The best ensembles of Siamese networks perform worse than eCNN, which is an ensemble of standard CNNs

3. Ensembling CNNs and Siamese networks together improves the performance of the ensemble of CNNs.

In Table 7.4, I report some more state-of-the-art results on the CAT and BIRD datasets. The accuracy of the ensembles described in this paper is competitive with those reported in the literature.

7.5 Discussion

The experiments presented in this paper show the robustness of the Siamese ensemble approach since the same method obtained very competitive results on two different datasets without the need of any specific hyperparameter tuning. Hence, this ensemble can be a good off-the-shelf method to classify animal sounds.

In this chapter, I tested and compared multiple methods to use Siamese networks for classification. I varied backbone networks and clustering options to see which methods works best, and in particular how their ensemble works. The ensemble of Siamese networks outperform the stand-alone Siamese networks.

In general, the methods based on Siamese networks seem to work worse than the ones based on CNNs. Siames ensembles do not outperform single networks either, since a simple VGG16 has better performance. However, in the literature, Siamese networks are usually opposed to classic CNNs and they are tested against them. In this chapter I showed that such different methods, that are likely to learn different features of the inputs, can be used together to improve the performance of both approaches. In this chapter, extensive tests have been performed on two medium size datasets. In order to have a deeper understanding of this problem, further testing would be required.

Chapter 8

Generalization of Covid-19 Classifiers

In this chapter I present other work related to image classification using Convolutional Networks. In particular, I introduce my work [220*] regarding the generalization of CNNs when it comes to Covid-19 classification from X-Ray images.

8.1 Covid-19 Classification

The publication of small datasets containing X-Ray and CT images of patients affected by COVID-19 led many researchers to use deep learning methods on these data [221, 222, 223, 224, 225]. It has already been shown that the chest X-Rays of people that suffered from Covid-19 related pneumonia could be distinguished from normal pneumonia e healthy lungs [226]. A lot of papers published on Covid-19 detection using X-Ray images reported a very high accuracy and a great performance in terms of recall and area under the ROC-curve (AUC) [222, 223, 224, 225]. However, Cohen et al. [227] warned the research community about the limits of the generalization ability of X-Ray images classificators, since a neural network might learn to classify the images of a specific dataset but fail to reproduce the same performance on other datasets. That paper was not the first one on the topic, but followed other works already published in the literature [228, 229, 230, 231, 232]. I tested if that was the case for many of the testing protocols used for Covid-19 classification.

I tested simple neural networks on four well-known datasets containing chest X-Ray images annotated by expert radiologists. These datasets are also among the most common ones used in the literature when it comes to Covid-19 classification. The idea is to check if a simple network can distinguish the source dataset of an image. This is relevant because most published and highly cited papers dealing with Covid-19 detection merged different datasets to train a neural network to recognize the ones of patients affected by Covid-19. Since, at that time, most Covid-19 images came from the same dataset and all the other images came from other data sources, if a network could only learn to tell the source dataset, it would also be able to use this information to perform well on a test set, but only because it is recognizing the source dataset. Hence, it would completely fail to generalize on new data.

In order to check this, I ran the same kind of experiments proposed by highly cited and published papers on the topic, but I covered all the centers of the images of the dataset, both in training and testing, with a large black square of fixed size that covered most of the lungs in the image and that made it impossibile to classify Covid on that corrupted dataset. I wanted to see if I were still able to classify the images. I plot some examples belonging to different datasets in Figure 8.1. I find that it is possible to recognize the dataset. Our trained AlexNet managed to obtain much better results than the ones reported by [233] on a standard pneumonia detection dataset. Hence, the testing protocols used in many papers in the literature are biased, since the classifiers might learn to recognize the source dataset and fail to generalize on other datasets.

8.2 Related Works

Chest X-Ray classification is a relevant topic in deep learning. There are multiple datasets proposed for chest X-Ray classification [233, 119]. Many neural networks have been trained on those datasets: Rajpurkar et al. [233] report a 0.76 ROC-AUC for the pneumonia detection task on the dataset proposed in [119]. Potential biases in chest X-Ray recognition has already studied before [227, 228, 229, 230, 234, 235]. Besides, Cohen et al. [227] questioned the real world applications of these classifiers, based on their robustness and generalization. They trained Densenet on a group of chest X-Ray datasets and tested it on a different one. All the datasets used in their analysis shared the same set of labels. When the first Covid-19 dataset [236] was shared by Cohen et al., deep learning researchers wanted to classify that dataset and created different test protocols by merging this dataset with older chest X-Ray image datasets proposed in the literature. I shall now list some of their testing protocols. Narin et al. [223] proposed a small dataset containing 50 COVID-19 cases, taken from the Cohen repository, and 50 healthy cases taken from Kaggle. They trained five ResNets, one for each fold of their cross-fold validation protocol, and obtained an average accuracy of 98%. Apostolopoulos et al. [225] also combined Cohen repository with other datasets to create a repository containing 224 images of COVID-19, 700 images with pneumonia and 504 healthy patients. They trained VGG-16 using a 10 fold cross validation and reahed 93.48% accuracy. Wang et al. [237] proposed a custom network called Covid-Net, a new architecture introduced in their paper. Their dataset is a little larger than the previous ones, since it containes 183 cases of COVID-19, but 5,538 of Pneumonia and 8,066 of healthy lungs. Their test set contains 100 images of pneumonia and of healthy lungs, but only 31 of COVID-19, since they are more rare. They explicitely mention in their paper that images of the same patient cannot be both in the training and in the test set, which is necessary for a good testing protocol in this field, but is not mentioned in the papers of the protocols that I described before. They obtained a 92% accuracy on this dataset.

Castiglioni et al. [238] proposed a completely different dataset collected in Italy and completely independent on Cohen's. Their training set contained 250 COVID-19 and 250 healthy images, and their test set contained 74 positive and 36 negative samples. They used an ensemble of 10 ResNet50 and obtained a ROC-AUC of 0.80, which is much worse than the other results that I mentioned above, but on a completely different dataset that does not suffer from the dataset merging problem that I wanted to study. The bad classification results obtained with an apparently safer testing protocol suggests that other results might be too optimistic. Tartaglione et al. [231] also used a private dataset, that they called CORDA. They manage to test their networks with two different sources of Covid-19 images, hence they managed to combine different datasets for training and for testing. They show that the accuracies of their classifiers range from random to 0.97 ROC-AUC depending on which source datasets are on the training and test dataset. Tabik et al. [232] collected a different dataset in Spain, named COVIDGR. Their annotations also differentiate COVID-19 cases depending on their severity. This dataset is divided into four different classes. Their model relies on GANs in order to prevent it to learn specific features that depend on the different source datasets. They reach accuracies of 97%, 88% and 66% for severe, moderate and mild cases. Interestingly, they could not detect asymptomatic cases from healthy patients.

8.3 Datasets

Here I shall introduce the four datasets used in the tests below, that are some of the most used datasets in this field.

NIH Dataset The ChestX-Ray8 dataset [119] was released by the National Institute of Health. This is why I call it NIH. This is a very large public dataset, containing 108,948 images of 32,717 different patients, classified into 8 different labels.

CHE dataset Irvin et al. shared Chexpert [239], a large dataset containing 224,316 chest radiographs of 65,240 patients labelled in 14 different

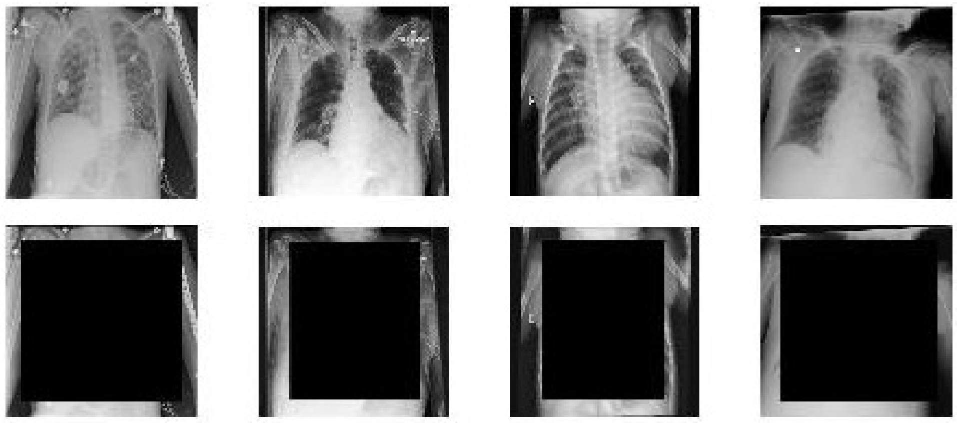


Figure 8.1: Original and transformed samples from the 4 datasets, 300 sized black square (Left to right: COV, NIH, CHE, KAG)

categories. I call this dataset CHE. Its test set is not publicly available, hence I used the validation set fo this purpose, since I so not use any data for validation in our tests.

KAG dataset This is a dataset shared by Dr. Paul Mooney [115], containing images of viral and bacterial pneumonia. It it made by 5,863 pediatric images, which means that the images it contains are very different from the others. I call this dataset KAG.

COV dataset Our source of COVID-19 images is the repository made available by Cohen et al. [236], which is the main source of many papers dealing with COVID-19. In the moment I are writing, it contains 144 images of frontal X-Ray images of patients potentially positive to COVID-19. Metadata are available for every sample, containing the patient ID and, most of the times, the location and other notes that contain the reference to the doctor that uploaded the images. I refer to this dataset as COV.

8.4 Methods

I ran two similar experiments with two different testing protocols. I combined the four datasets that I introduced above in different ways in order to understand which combinations allow classifiers to recognize the source dataset. I preprocessed the images by resizing them as follows: at first, I want their smallest dimension to be equal to 360, then I cover the center of the images with a square of different sizes, depending on the experiment. After that, I resized them to 227x227 in order to be used as inputs on AlexNet.

In all our experiments I managed to cover at least most of the lungs with our squares, but leaving information about the source dataset on the boudaries.

I divided the three datasets that do not contain Covid samples into training and test sets as they were already divided. Then, the COV dataset was split into folds in two different ways that I shall now explain. The former (PAT-OUT) requires that all the samples relative to the same patients are in the same folds. The latter (DOC-OUT) also requires that scans uploaded by the same doctor are in the same folds. In this datasets, all the scans of a given patient are uploaded by the same doctor, so the second protocol is stricter than the first. The reason behind this is that those scans might share common features due to the fact that they might have been taken using the same X-Ray machine or share the same preprocessing.

The first experiment consists in merging the training sets of all the datasets and tried to recognize the source dataset of the images, which means that there are 4 different labels. In this experiment I used the DOC-OUT protocol. I call this experiment dataset classification.

In the second experiment, I used a protocol that is inspired by the one proposed by Cohen in [227]. I split this experiment into three parts. In the first part, I leave the NIH dataset out to be used in the test set. Then I use a cross fold validation protocol to train AlexNet on CHE, KAG and all but one fold of COV, and test the network on NIH and the left out fold. I then iterate this procedure on folds and I do this for the three non-Covid dataset. In this case, I used both the DOC-OUT and PAT-OUT protocol. I call this COV recognition experiment.

8.5 Results and Discussion

Classification performance I test our AlexNet on class vs. class environment, using AUC as our metric. This is because the dataset is heavily unbalanced. In Table 8.5 I show the results for the dataset recognition problem. The reader can easily see that the problem can be easily solved by AlexNet even if the images were corrupted by our preprocessing.

	Datasets					
Square size	COV-NIH	COV-CHE	COV-KAG	NIH-CHE	NIH-KAG	CHE-KAG
270	0.921	0.965	0.990	0.996	0.999	0.999
300	0.928	0.987	0.993	0.999	0.999	0.999

Table 8.1: ROC-AUC of the dataset classification task.

In Table 8.5 I calculate the confusion matrix of this method. It seems that COV images are the only ones that are often misclassified, and this suggested us that maybe those images do not share any particular feature, but the other datasets do. The COV dataset was collected by a large number

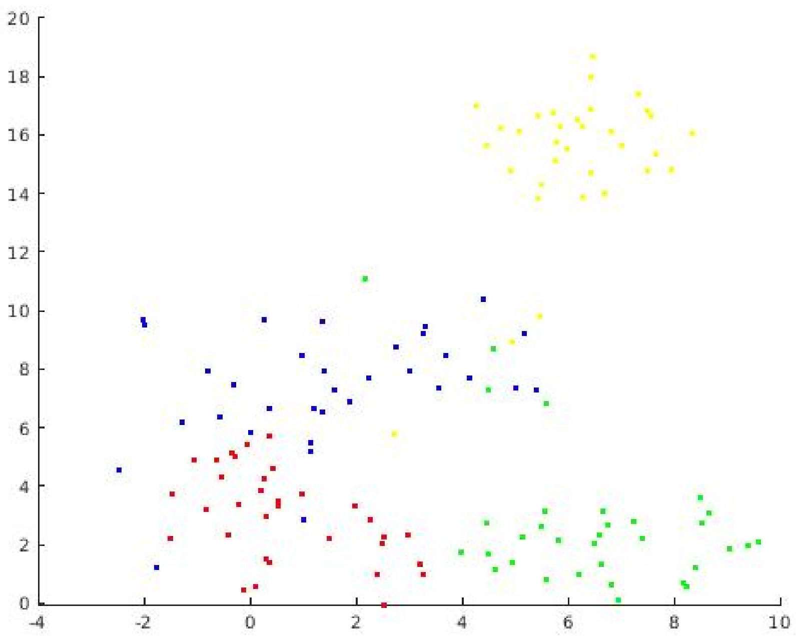


Figure 8.2: t-SNE for dataset classification. Blue is COV, green is CHE, red is NIH and yellow is KAG.

of different doctors worldwide, so the images do not necessarily have to share any feature.

In Table 8.3 I report the AUC of the other experiment, which the COV

		Dataset			
Dataset	Test size	NIH	CHE	KAG	COV
NIH	114576	106911 (93.3%)	4197 (3.7%)	34 (0.0%)	3434 (3%)
CHE	363	9 (2.5%)	342 (94.2%)	1 (0.3%)	11 (3.0%)
KAG	6864	53 (0.8%)	131 (1.9%)	6474 (94.3%)	206 (3.0%)
COV	144	33 (23%)	13 (7.7%)	10 (7.0%)	88 (62.0%)

Table 8.2: Confusion matrix for dataset classification.

recognition task. In this case, I cannot recognize the left-out dataset with our netowrk. Our interpretation is that there are not many inter-dataset features shared among the three non Covid datasets, besides that there are not many intra-dataset features shared by the samples in the the COV dataset. I use t-distributed stochastic neighbor embbeddings [240] to visualize this more specifically in Figure 8.5.

COV points have overlaps with the other datasets, that in general form very recognizable cluster. The four clusters there help to understand how the classifier works if one of the non Covid datasets is left out. If, for example, the yellow dataset is left out and a classifier is trained on those data, every point above the blue cluster will be classified as COV point with very high confidence, even higher that blue points themselves, since it would be very far from green and red points. This is what leads AlexNet to perform even worse than random when the non Covid point in the test set are the yellow points.

Table 8.3: ROC-AUC of the COV recognition task.

	Dataset		
Test protocol	Leave-out NIH	Leave-out CHE	Leave-out KAG
DOC-OUT	0.68	0.62	0.36
PAT-OUT	0.68	0.62	0.42

Conclusions The results of our experiments are very clear. If a classifier is trained on a dataset obtained by merging different X-Ray datasets, there is a very high chance that this classifier learns to recognize the source dataset rather than any evidence of illness. I showed it by eliminating all the features of the images that could reasonably suggest the presence of Covid-19 pneumonia in the inputs.

However, if the training protocol takes into account this bias and differentiate the training and test datasets, this problem does not happen anymore. However, the desired situation would be that data points of the different datasets form two clusters in a feature space that are indistinguishible, but this does not happen and lower AUCs are due to the fact that test points are very different from every other sample seen in the training set.

This shows a potential bias that explains the results of most of the papers cited in the Related Works section without requiring the classifiers trained in those papers actually recognize Covid and would actually be useful on a different dataset. Hence, I stress the importance of collecting diverse data from multiple sources to train classifiers in this field. Besides, an evaluation protocol should always use different datasets for testing and for training to prove the robustness and the generalization power of the model.

Chapter 9

Discussion

In this thesis, I extensively studied and tested methods to create ensembles of neural networks. I investigated a field which has been relevant in artificial intelligence and in deep learning for a long time, but that lacks the large amount of experiments that most classical artificial intellingence methods have.

The most relevant contribution of this thesis consists in the testing protocols to evaluate the effectiveness of the ensemble methods. I compared the methods proposed here with other ensemble methods of similar cardinality, with other classifiers with similar computational requirements or with state-of-the-art methods published in the literature. Most of the literature published in this field did not primarily focus on which ensemble method really was effective. There is a fair chance that most ensembles proposed in the literature would not have outperformed naive ensembles made by the same network trained multiple times. At least, in those papers there is no evidence of that. Those papers often focused on creating a strong classifier on a given dataset, so the authors trained multiple models and only reported that the ensemble was the the classifier that obtained the best results.

In this thesis I tested ensemble models on a variety of datasets that differ in size and fields. I showed that ensemble models proposed here outperform the baseline models on a variety of tasks. In Chapter 4 I used them to classify naural images of insect pests for crop damage prevention. In the same chapter, the same methods were applied to spectrograms extracted from audio recordings. In the next chapters, I focused on medium sized biomedical datasets, which contain images extracted with different methods (e.g: X-rays, endoscopies, microscopes...). Hence, I showed that ensemble methods work over a large variety of fields.

The effectiveness of ensemble models was already proved in the literature and has been known for a long time. Many papers that report state-of-the-art results use ensembles of multiple networks to boost the performance of their methods as much as they can. Besides, theoretical analysis of why ensemble models outperform single ones had already been published, although there is not much work on how much an ensemble model can improve the accuracy of a single network and in which situations ensembles work best. In this thesis we saw that the improvement of ensemble models over single networks can be seen on varoius datasets and on various methods. We did not find strong evidence that suggests which frameworks are the ones that mostly benifit from ensembles. This work suggests that ensembles are more effective on small datasets. This is suggested in particular by the analysis of the results of Section 4.1, where two different pest datasets are used. The performance improvement given by the ensemble is more relevant in the small dataset.

The relevance of testing protocols is the glue that keeps this thesis together and it is also the topic of Chapter 8, that deals with misleading excellent results in classifying Covid-19 using x-ray images. In Chapter 8 I show how most papers among the early published ones after the outbreak of Covid-19 obtained amazing results just by recognizing features of the input data that did not depend on Covid-19.

As far as it concerns the topic of deep ensembles, the most relevant finding of this thesis are the importance of diversity among the deep classifiers to create a strong ensemble method and multiple ways to obtain that diversity. I applied different techniques in different fields and using different datasets, ranging from natural images to spectrograms obtained from audio signals. Hence, a direct comparisons among the methods that I propose here cannot be done, but that was not the aim of this work. There is no way to directly compare the results of two ensemble methods that are tested on different datasets. In general, all the ensemble methods that I propose share the following properties:

1. They all outperform naive ensembles of the same network trained multiple times;
2. They are very general methods that can be applied to any neural network;
3. They are easy to implement and can be used as off-the-shelf methods to improve the performance of a general neural network.

A relevant feature that seems to hold for all the experiments is the need of not deteriorating the performance of the single models in the ensemble to obtain a good final results. Although this might seem obvious, it is also not as true as one could think. From the experiments of Section 4.2, there is evidence that in all the three datasets the ensemble of all the networks performs worse than the ensemble of the networks trained with the Super augmentation protocols. Those networks performance is better than the one of the other networks in the ensemble, hence they reach a better performance when they are the only ones in the ensemble. However, the best ensemble that we created in that section contains all the networks trained with different methods to extract a spectrogram, whose performance was generally lower than the original one that I used. Hence, it is very hard to predict when the ensemble of networks with different performance can improve the overall performance or not.

In Chapter 5, changing the activation functions of the networks does not lower their performance. In that case, the large the ensemble, the better the network performs. This principle is never violated throught this thesis, although it was quite expected.

In general, ensemble models seem to have a larger improvement in small and medium sized dataset with respect to the single networks. This observation leads me to suppose that this is due to the overfitting of a single model on a small dataset. Ensemble models are made by different networks that converge in different local minima. These minima are known to be more different in case of small datasets. Hence, averaging the results of networks trained on small datasets might lead to average out the component of the prediction that is due to the overfitting, while mantaining the meaningful component of the score. Besides, this hypothesis is consistent with the fact that diversity is relevant in ensemble models. Training a model fixed size on a small dataset can lead to overfitting. However, overfitting often leads to different results on unseen data in independent training rounds. Hence, the networks predictions are probably very diverse on unseen data when the training set is small.

Chapter 10

Bibliography

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, 2017.

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[5] J. W. Tukey *et al.*, *Exploratory data analysis*, vol. 2. Reading, Mass., 1977.

[6] B. V. Dasarathy and B. V. Sheela, “A composite classifier system design: Concepts and methodology,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[7] L. K. Hansen and P. Salamon, “Neural network ensembles,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[8] R. E. Schapire, “The strength of weak learnability,” *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[9] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[10] P. S. A. Krogh *et al.*, "Learning with ensembles: How over-fitting can be useful," in *Proceedings of the 1995 Conference*, vol. 8, p. 190, 1996.

[11] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Information fusion*, vol. 6, no. 1, pp. 5–20, 2005.

[12] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[13] E. Menahem, A. Shabtai, L. Rokach, and Y. Elovici, "Improving malware detection by applying multi-inducer ensemble," *Computational Statistics & Data Analysis*, vol. 53, no. 4, pp. 1483–1494, 2009.

[14] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[15] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Information Fusion*, vol. 52, pp. 1–12, 2019.

[16] J. J. G. Adeva, U. Beresi, and R. Calvo, "Accuracy and diversity in ensembles of text categorisers," *CLEI Electronic Journal*, vol. 9, no. 1, pp. 1–12, 2005.

[17] M. Gashler, C. Giraud-Carrier, and T. Martinez, "Decision tree ensemble: Small heterogeneous is better than large homogeneous," in *2008 Seventh International Conference on Machine Learning and Applications*, pp. 900–905, IEEE, 2008.

[18] I. Sutskever and V. Nair, "Mimicking go experts with convolutional neural networks," in *International Conference on Artificial Neural Networks*, pp. 101–110, Springer, 2008.

[19] L. Nanni, A. Lumini, and S. Ghidoni, "Ensemble of deep learned features for melanoma classification," *arXiv preprint arXiv:1807.08008*, 2018.

[20] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[21] R. Paul, L. Hall, D. Goldgof, M. Schabath, and R. Gillies, "Predicting nodule malignancy using a cnn ensemble approach," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[22] S. H. Kassani, P. H. Kassani, M. J. Wesolowski, K. A. Schneider, and R. Deters, "Classification of histopathological biopsy images using ensemble of deep learning networks," *arXiv preprint arXiv:1909.11870*, 2019.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[25] S. Qummar, F. G. Khan, S. Shah, A. Khan, S. Shamshirband, Z. U. Rehman, I. A. Khan, and W. Jadoon, "A deep learning ensemble approach for diabetic retinopathy detection," *IEEE Access*, vol. 7, pp. 150530–150539, 2019.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[29] K. Liu, M. Zhang, and Z. Pan, "Facial expression recognition with cnn ensemble," in *2016 international conference on cyberworlds (CW)*, pp. 163–166, IEEE, 2016.

[30] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, *et al.*, "Challenges in representation learning: A report on three machine learning contests," in *International conference on neural information processing*, pp. 117–124, Springer, 2013.

[31] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 31–40, 2016.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[33] A. Gilbert, L. Piras, J. Wang, F. Yan, A. Ramisa, E. Dellandrea, R. J. Gaizauskas, M. Villegas, and K. Mikolajczyk, "Overview of the imageclef 2016 scalable concept image annotation task.," in *CLEF (Working Notes)*, pp. 254–278, 2016.

[34] P. Pandey, A. Deepthi, B. Mandal, and N. B. Puhan, "Foodnet: Recognizing foods using ensemble of deep networks," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1758–1762, 2017.

[35] H.-K. Huang, C.-F. Chiu, C.-H. Kuo, Y.-C. Wu, N. N. Chu, and P.-C. Chang, "Mixture of deep cnn-based ensemble model for image retrieval," in *2016 IEEE 5th Global Conference on Consumer Electronics*, pp. 1–2, IEEE, 2016.

[36] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[37] L. Nanni, S. Ghidoni, and S. Brahnam, "Ensemble of convolutional neural networks for bioimage classification," *Applied Computing and Informatics*, 2020.

[38] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[39] B. Zhang, S. Qi, P. Monkam, C. Li, F. Yang, Y.-D. Yao, and W. Qian, "Ensemble learners of multiple deep cnns for pulmonary nodules classification using ct images," *IEEE Access*, vol. 7, pp. 110358–110371, 2019.

[40] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[41] D. H. Wolpert and W. G. Macready, "An efficient method to estimate bagging's generalization error," *Machine Learning*, vol. 35, no. 1, pp. 41–55, 1999.

[42] E. Bauer and R. Kohavi, “An empirical comparison of voting classification algorithms: Bagging, boosting, and variants,” *Machine learning*, vol. 36, no. 1, pp. 105–139, 1999.

[43] P.-K. Kim and K.-T. Lim, “Vehicle type classification using bagging and convolutional neural network on multi view surveillance image,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 41–46, 2017.

[44] X. Dong, L. Qian, and L. Huang, “A cnn based bagging learning approach to short-term load forecasting in smart grid,” in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1–6, IEEE, 2017.

[45] J. Guo and S. Gould, “Deep cnn ensemble with data augmentation for object detection,” *arXiv preprint arXiv:1506.07224*, 2015.

[46] B. Savelli, A. Bria, M. Molinara, C. Marrocco, and F. Tortorella, “A multi-context cnn ensemble for small lesion detection,” *Artificial intelligence in medicine*, vol. 103, p. 101749, 2020.

[47] Y. Chen, Y. Wang, Y. Gu, X. He, P. Ghamisi, and X. Jia, “Deep learning ensemble for hyperspectral image classification,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 6, pp. 1882–1897, 2019.

[48] Y. Fan, J. C. Lam, and V. O. Li, “Multi-region ensemble convolutional neural network for facial expression recognition,” in *International Conference on Artificial Neural Networks*, pp. 84–94, Springer, 2018.

[49] Y. Gan, J. Chen, and L. Xu, “Facial expression recognition boosted by soft label with a diverse ensemble,” *Pattern Recognition Letters*, vol. 125, pp. 105–112, 2019.

[50] G. Antipov, S.-A. Berrani, and J.-L. Dugelay, “Minimalistic cnn-based ensemble model for gender prediction from face images,” *Pattern recognition letters*, vol. 70, pp. 59–65, 2016.

[51] C. Ju, A. Bibaut, and M. van der Laan, “The relative performance of ensemble methods with deep convolutional neural networks for image classification,” *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018.

[52] B. Harangi, “Skin lesion classification with ensembles of deep convolutional neural networks,” *Journal of biomedical informatics*, vol. 86, pp. 25–32, 2018.

[53] M. Lyksborg, O. Puonti, M. Agn, and R. Larsen, “An ensemble of 2d convolutional neural networks for tumor segmentation,” in *Scandinavian Conference on Image Analysis*, pp. 201–211, Springer, 2015.

[54] R. Minetto, M. P. Segundo, and S. Sarkar, “Hydra: An ensemble of convolutional neural networks for geospatial land classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 6530–6541, 2019.

[55] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, “A survey on ensemble learning,” *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020.

[56] A. Lumini, L. Nanni, and G. Maguolo, “Deep learning for plankton and coral classification,” *Applied Computing and Informatics*, 2020.

[57] R. E. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Machine learning*, vol. 37, no. 3, pp. 297–336, 1999.

[58] A. Taherkhani, G. Cosma, and T. M. McGinnity, “Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning,” *Neurocomputing*, vol. 404, pp. 351–366, 2020.

[59] C. Potes, S. Parvaneh, A. Rahman, and B. Conroy, “Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds,” in *2016 computing in cardiology conference (CinC)*, pp. 621–624, IEEE, 2016.

[60] C. Gao, P. Li, Y. Zhang, J. Liu, and L. Wang, “People counting based on head detection combining adaboost and cnn in crowded surveillance environment,” *Neurocomputing*, vol. 208, pp. 108–116, 2016.

[61] W. Chen, Q. Sun, J. Wang, J.-J. Dong, and C. Xu, “A novel model based on adaboost and deep cnn for vehicle classification,” *Ieee Access*, vol. 6, pp. 60445–60455, 2018.

[62] X. Zhou, L. Xie, P. Zhang, and Y. Zhang, “An ensemble of deep neural networks for object tracking,” in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 843–847, IEEE, 2014.

[63] G. Brown, J. L. Wyatt, P. Tino, and Y. Bengio, “Managing diversity in regression ensembles.,” *Journal of machine learning research*, vol. 6, no. 9, 2005.

[64] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 417–425, 1998.

[65] D. W. Opitz, J. W. Shavlik, *et al.*, "Generating accurate and diverse members of a neural-network ensemble," *Advances in neural information processing systems*, pp. 535–541, 1996.

[66] D. Wang and M. Alhamdoosh, "Evolutionary extreme learning machine ensembles with size control," *Neurocomputing*, vol. 102, pp. 98–110, 2013.

[67] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[68] B. E. Rosen, "Ensemble learning using decorrelated neural networks," *Connection science*, vol. 8, no. 3-4, pp. 373–384, 1996.

[69] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[70] G. Giacinto and F. Roli, "Design of effective neural network ensembles for image classification purposes," *Image and Vision Computing*, vol. 19, no. 9-10, pp. 699–707, 2001.

[71] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Information Sciences*, vol. 264, pp. 104–117, 2014.

[72] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1962–1979, 2009.

[73] S. Kariyappa and M. K. Qureshi, "Improving adversarial robustness of ensembles with diversity training," *arXiv preprint arXiv:1901.09981*, 2019.

[74] N. Dvornik, C. Schmid, and J. Mairal, "Diversity with cooperation: Ensemble methods for few-shot classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3723–3731, 2019.

[75] G. U. Yule, "Vii. on the association of attributes in statistics: with illustrations from the material of the childhood society, &c," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 194, no. 252-261, pp. 257–319, 1900.

[76] P. H. Sneath, R. R. Sokal, *et al.*, *Numerical taxonomy. The principles and practice of numerical classification.* 1973.

[77] D. B. Skalak *et al.*, "The sources of increased accuracy for two proposed boosting algorithms," in *Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, vol. 1129, p. 1133, Citeseer, 1996.

[78] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[79] G. Giacinto and F. Roli, "An approach to the automatic design of multiple classifier systems," *Pattern recognition letters*, vol. 22, no. 1, pp. 25–33, 2001.

[80] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 330–345, Springer, 2012.

[81] G. D. Cavalcanti, L. S. Oliveira, T. J. Moura, and G. V. Carvalho, "Combining diversity measures for ensemble pruning," *Pattern Recognition Letters*, vol. 74, pp. 38–45, 2016.

[82] L. Wen, L. Gao, and X. Li, "A new snapshot ensemble convolutional neural network for fault diagnosis," *Ieee Access*, vol. 7, pp. 32037–32047, 2019.

[83] W. Zhang, J. Jiang, Y. Shao, and B. Cui, "Snapshot boosting: a fast ensemble framework for deep neural networks," *Science China Information Sciences*, vol. 63, no. 1, pp. 1–12, 2020.

[84] W. Chen, X. Li, L. Gao, and W. Shen, "Improving computer-aided cervical cells classification using transfer learning based snapshot ensemble," *Applied Sciences*, vol. 10, no. 20, p. 7292, 2020.

[85] A. Mehrtash, P. Abolmaesumi, P. Golland, T. Kapur, D. Wassermann, and W. M. Wells III, "Pep: Parameter ensembling by perturbation," *arXiv preprint arXiv:2010.12721*, 2020.

[86] S. Masoudnia, O. Mersa, B. N. Araabi, A.-H. Vahabie, M. A. Sadeghi, and M. N. Ahmadabadi, "Multi-representational learning for offline signature verification using multi-loss snapshot ensemble of cnns," *Expert Systems with Applications*, vol. 133, pp. 317–330, 2019.

[87] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[88] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

[89] Z. Chen, J. Duan, L. Kang, and G. Qiu, "Class-imbalanced deep learning via a class-balanced ensemble," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[90] B. Fielding, T. Lawrence, and L. Zhang, "Evolving and ensembling deep cnn architectures for image classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[91] S. Lee, S. Purushwalkam, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra, "Stochastic multiple choice learning for training diverse deep ensembles," *arXiv preprint arXiv:1606.07839*, 2016.

[92] A. Dutt, D. Pellerin, and G. Quénot, "Coupled ensembles of neural networks," *Neurocomputing*, vol. 396, pp. 346–357, 2020.

[93] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[94] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[95] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European conference on computer vision*, pp. 87–102, Springer, 2016.

[96] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pp. 1988–1996, 2014.

[97] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation," in *European conference on computer vision*, pp. 566–579, Springer, 2012.

[98] C. Ding and D. Tao, "Robust face recognition via multimodal deep face representation," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2049–2058, 2015.

[99] I. Masi, S. Rawls, G. Medioni, and P. Natarajan, "Pose-aware face recognition in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4838–4846, 2016.

[100] Y. Wu, H. Liu, and Y. Fu, "Low-shot face recognition with hybrid classifiers," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1933–1939, 2017.

[101] J. Y. Choi and B. Lee, "Ensemble of deep convolutional neural networks with gabor face representations for face recognition," *IEEE Transactions on Image Processing*, vol. 29, pp. 3270–3281, 2019.

[102] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[103] B. Harangi, A. Baran, and A. Hajdu, "Classification of skin lesions using an ensemble of deep neural networks," in *2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pp. 2575–2578, IEEE, 2018.

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[105] A. Mahbod, G. Schaefer, C. Wang, G. Dorffner, R. Ecker, and I. Ellinger, "Transfer learning using a multi-scale and multi-network ensemble for skin lesion classification," *Computer methods and programs in biomedicine*, vol. 193, p. 105475, 2020.

[106] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, PMLR, 2019.

[107] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[108] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, *et al.*, "Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)," *arXiv preprint arXiv:1902.03368*, 2019.

[109] S. A. A. Ahmed, B. Yanikoğlu, Ö. Göksu, and E. Aptoula, "Skin lesion classification with deep cnn ensembles," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2020.

[110] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pp. 8697–8710, 2018.

[111] N. Gessert, M. Nielsen, M. Shaikh, R. Werner, and A. Schlaefer, “Skin lesion classification using ensembles of multi-resolution efficientnets with meta data,” *MethodsX*, vol. 7, p. 100864, 2020.

[112] M. F. Hashmi, S. Katiyar, A. G. Keskar, N. D. Bokde, and Z. W. Geem, “Efficient pneumonia detection in chest xray images using deep transfer learning,” *Diagnostics*, vol. 10, no. 6, p. 417, 2020.

[113] D. Kermany, K. Zhang, and M. Goldbaum, “Large dataset of labeled optical coherence tomography (oct) and chest x-ray images,” *Mendeley Data, v3 http://dx. doi. org/10.17632/rscbjbr9sj*, vol. 3, 2018.

[114] T. Rahman, M. E. Chowdhury, A. Khandakar, K. R. Islam, K. F. Islam, Z. B. Mahbub, M. A. Kadir, and S. Kashem, “Transfer learning with deep convolutional neural network (cnn) for pneumonia detection using chest x-ray,” *Applied Sciences*, vol. 10, no. 9, p. 3233, 2020.

[115] M. Paul, “chest x-ray images (pneumonia).” https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/version/2, 2017.

[116] V. Chouhan, S. K. Singh, A. Khamparia, D. Gupta, P. Tiwari, C. Moreira, R. Damaševičius, and V. H. C. De Albuquerque, “A novel transfer learning based approach for pneumonia detection in chest x-ray images,” *Applied Sciences*, vol. 10, no. 2, p. 559, 2020.

[117] J. R. Ferreira, D. A. C. Cardenas, R. A. Moreno, M. d. F. de Sá Rebelo, J. E. Krieger, and M. A. Gutierrez, “Multi-view ensemble convolutional neural network to improve classification of pneumonia in low contrast chest x-ray images,” in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1238–1241, IEEE, 2020.

[118] A. Gooßen, H. Deshpande, T. Harder, E. Schwab, I. Baltruschat, T. Mabotuwana, N. Cross, and A. Saalbach, “Deep learning for pneumothorax detection and localization in chest radiographs,” *arXiv preprint arXiv:1907.07324*, 2019.

[119] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, “Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106, 2017.

[120] F. Ahmad, A. Farooq, and M. U. Ghani, "Deep ensemble model for classification of novel coronavirus in chest x-ray images," *Computational Intelligence and Neuroscience*, vol. 2021, 2021.

[121] A. Afifi, N. E. Hafsa, M. A. Ali, A. Alhumam, and S. Alsalman, "An ensemble of global and local-attention based convolutional neural networks for covid-19 diagnosis on chest x-ray images," *Symmetry*, vol. 13, no. 1, p. 113, 2021.

[122] S. D. Deb and R. K. Jha, "Covid-19 detection from chest x-ray images using ensemble of cnn models," in *2020 International Conference on Power, Instrumentation, Control and Computing (PICC)*, pp. 1–5, IEEE, 2020.

[123] E. Vantaggiato, E. Paladini, F. Bougourzi, C. Distante, A. Hadid, and A. Taleb-Ahmed, "Covid-19 recognition using ensemble-cnns in two new chest x-ray databases," *Sensors*, vol. 21, no. 5, p. 1742, 2021.

[124] J. Kang, Z. Ullah, and J. Gwak, "Mri-based brain tumor classification using ensemble of deep features and machine learning classifiers," *Sensors*, vol. 21, no. 6, p. 2222, 2021.

[125] J. Islam and Y. Zhang, "Brain mri analysis for alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks," *Brain informatics*, vol. 5, no. 2, pp. 1–14, 2018.

[126] D. Pan, A. Zeng, L. Jia, Y. Huang, T. Frizzell, and X. Song, "Early detection of alzheimer's disease using magnetic resonance imaging: A novel approach combining convolutional neural networks and ensemble learning," *Frontiers in neuroscience*, vol. 14, 2020.

[127] M. Leming, J. M. Górriz, and J. Suckling, "Ensemble deep learning on large, mixed-site fmri datasets in autism and other tasks," *arXiv preprint arXiv:2002.07874*, 2020.

[128] R. Rasti, M. Teshnehlab, and S. L. Phung, "Breast cancer diagnosis in dce-mri using mixture ensemble of convolutional neural networks," *Pattern Recognition*, vol. 72, pp. 381–390, 2017.

[129] S. Lee, S. J. Son, J. Oh, and N. Kwak, "Handwritten music symbol classification using deep convolutional neural networks," in *2016 International Conference on Information Science and Security (ICISS)*, pp. 1–5, IEEE, 2016.

[130] M. Awni, M. I. Khalil, and H. M. Abbas, "Deep-learning ensemble for offline arabic handwritten words recognition," in *2019 14th International Conference on Computer Engineering and Systems (ICCES)*, pp. 40–45, IEEE, 2019.

[131] B. Alizadehashraf and S. Roohi, "Persian handwritten character recognition using convolutional neural network," in *2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pp. 247–251, IEEE, 2017.

[132] Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten chinese character recognition using googlenet and directional feature maps," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 846–850, IEEE, 2015.

[133] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking cnn models for audio classification," *arXiv preprint arXiv:2007.11154*, 2020.

[134] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2015.

[135] S. Kahl, T. Wilhelm-Stein, H. Hussein, H. Klinck, D. Kowerko, M. Ritter, and M. Eibl, "Large-scale bird sound classification using convolutional neural networks.," in *CLEF (working notes)*, vol. 1866, 2017.

[136] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W.-P. Vellinga, J.-C. Lombardo, R. Planqué, S. Palazzo, and H. Müller, "Lifeclef 2017 lab overview: multimedia species identification challenges," in *International Conference of the Cross-Language Evaluation Forum for European Languages*, pp. 255–274, Springer, 2017.

[137] S. Zimmerman, U. Kruschwitz, and C. Fox, "Improving hate speech detection with deep learning ensembles," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[138] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Fifteenth annual conference of the international speech communication association*, 2014.

[139] P. Lopez-Meyer, J. A. del Hoyo Ontiveros, G. Stemmer, L. Nachman, and J. Huang, "Ensemble of convolutional neural networks for the dcase 2020 acoustic scene classification challenge," in *DCASE Workshop*, 2020.

[140] J. J. Huang and J. J. A. Leanos, "Aclnet: efficient end-to-end audio classification cnn," *arXiv preprint arXiv:1811.06669*, 2018.

[141] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions," *arXiv preprint arXiv:2005.14623*, 2020.

[142] H. Yang, C. Shi, and H. Li, “Acoustic scene classification using cnn ensembles and primary ambient extraction,” *the 2019 Challenge on Detection and Classification of Acoustic Scenes and Events*, 2019.

[143] J. Huang, H. Lu, P. Lopez Meyer, H. Cordourier, and J. Del Hoyo Ontiveros, “Acoustic scene classification using deep learning-based ensemble averaging,” 2019.

[144] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, IEEE, 2017.

[145] Y. Sakashita and M. Aono, “Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions,” *Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge*, 2018.

[146] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, and R. Tachibana, “Domestic activities classification based on cnn using shuffling and mixing data augmentation,” *DCASE 2018 Challenge*, 2018.

[147] F. Noman, C.-M. Ting, S.-H. Salleh, and H. Ombao, “Short-segment heart sound classification using an ensemble of deep convolutional neural networks,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1318–1322, IEEE, 2019.

[148] A. I. Humayun, M. Khan, S. Ghaffarzadegan, Z. Feng, T. Hasan, *et al.*, “An ensemble of transfer, semi-supervised and supervised learning methods for pathological heart sound classification,” *arXiv preprint arXiv:1806.06506*, 2018.

[149] L. Nanni, S. Brahnam, and G. Maguolo, “Data augmentation for building an ensemble of convolutional neural networks,” in *Innovation in Medicine and Healthcare Systems, and Multimedia*, pp. 61–69, Springer, 2019.

[150] M. V. Boland and R. F. Murphy, “A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells,” *Bioinformatics*, vol. 17, no. 12, pp. 1213–1223, 2001.

[151] L. Shamir, N. Orlov, D. M. Eckley, T. J. Macura, and I. G. Goldberg, “Iicbu 2008: a proposed benchmark suite for biological image anal-

ysis," *Medical & biological engineering & computing*, vol. 46, no. 9, pp. 943–947, 2008.

[152] L. Nanni, G. Maguolo, and F. Pancino, "Insect pest image detection and recognition based on bio-inspired methods," *Ecological Informatics*, vol. 57, p. 101089, 2020.

[153] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1254–1259, 1998.

[154] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[155] L. Deng, Y. Wang, Z. Han, and R. Yu, "Research on insect pest image detection and recognition based on bio-inspired methods," *Biosystems Engineering*, vol. 169, pp. 139–148, 2018.

[156] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8787–8796, 2019.

[157] F. Ren, W. Liu, and G. Wu, "Feature reuse residual networks for insect pest recognition," *IEEE Access*, vol. 7, pp. 122758–122768, 2019.

[158] J. Padmanabhan and M. J. Johnson Premkumar, "Machine learning in automatic speech recognition: A survey," *IETE Technical Review*, vol. 32, no. 4, pp. 240–251, 2015.

[159] L. Nanni, Y. M. Costa, D. R. Lucio, C. N. Silla Jr, and S. Brahnam, "Combining visual and acoustic features for audio classification tasks," *Pattern Recognition Letters*, vol. 88, pp. 49–56, 2017.

[160] S. K. Sahoo, T. Choubisa, and S. M. Prasanna, "Multimodal biometric person authentication: A review," *IETE Technical Review*, vol. 29, no. 1, pp. 54–75, 2012.

[161] Z. Cao, J. C. Principe, B. Ouyang, F. Dalgleish, and A. Vuorenkoski, "Marine animal classification using combined cnn and hand-designed image features," in *OCEANS 2015-MTS/IEEE Washington*, pp. 1–6, IEEE, 2015.

[162] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

[163] H. B. Sailor, D. M. Agrawal, and H. A. Patil, "Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification.," in *INTERSPEECH*, pp. 3107–3111, 2017.

[164] S.-h. Zhang, Z. Zhao, Z.-y. Xu, K. Bellisario, and B. C. Pijanowski, "Automatic bird vocalization identification based on fusion of spectral pattern and texture features," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 271–275, IEEE, 2018.

[165] D. M. Agrawal, H. B. Sailor, M. H. Soni, and H. A. Patil, "Novel teo-based gammatone features for environmental sound classification," in *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1809–1813, IEEE, 2017.

[166] X. Li, V. Chebiyyam, and K. Kirchhoff, "Multi-stream network with temporal attention for environmental sound classification," *arXiv preprint arXiv:1901.08608*, 2019.

[167] J. Sharma, O.-C. Granmo, and M. Goodwin, "Environment sound classification using multiple feature channels and deep convolutional neural networks," *arXiv preprint arXiv:1908.11219*, 2019.

[168] M. Mohaimenuzzaman, C. Bergmeir, I. T. West, and B. Meyer, "Environmental sound classification on the edge: Deep acoustic networks for extremely resource-constrained devices," *arXiv e-prints*, pp. arXiv–2103, 2021.

[169] J. Chaki, "Pattern analysis based acoustic signal processing: a survey of the state-of-art," *International Journal of Speech Technology*, pp. 1–43, 2020.

[170] L. Nanni, G. Maguolo, S. Brahnam, and M. Paci, "An ensemble of convolutional neural networks for audio classification," *Applied Sciences*, vol. 11, no. 13, p. 5796, 2021.

[171] M. Lasseck, "Audio-based bird species identification with deep convolutional neural networks.," in *CLEF (Working Notes)*, 2018.

[172] E. Sprengel, M. Jaggi, Y. Kilcher, and T. Hofmann, "Audio based bird species identification using deep learning techniques," tech. rep., 2016.

[173] G. Maguolo, M. Paci, and L. Nanni, "Audiogmenter: a matlab toolbox for audio data augmentation," *arXiv preprint arXiv:1912.05472*, 2019.

[174] Z. Prusa *et al.*, "The large time frequency analysis toolbox: Wavelets," 2013.

[175] L. Rabiner and R. Schafer, *Theory and applications of digital speech processing*. Prentice Hall Press, 2010.

[176] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[177] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics*, p. 101084, 2020.

[178] L. Nanni, Y. M. Costa, A. Lumini, M. Y. Kim, and S. R. Baek, "Combining visual and acoustic features for music genre classification," *Expert Systems with Applications*, vol. 45, pp. 108–117, 2016.

[179] Z. Zhao, S.-h. Zhang, Z.-y. Xu, K. Bellisario, N.-h. Dai, H. Omrani, and B. C. Pijanowski, "Automated bird acoustic event detection and robust species classification," *Ecological Informatics*, vol. 39, pp. 99–108, 2017.

[180] Y. Pandeya, D. Kim, and J. Lee, "Domestic cat sound classification using learned features from deep neural nets," *Applied Sciences*, vol. 8, no. 10, p. 1949, 2018.

[181] Y. R. Pandeya and J. Lee, "Domestic cat sound classification using transfer learning," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 2, pp. 154–160, 2018.

[182] A. Kumar and V. Ithapu, "A sequential self teaching approach for improving generalization in sound event recognition," in *International Conference on Machine Learning*, pp. 5447–5457, PMLR, 2020.

[183] J. Kim, "Urban sound tagging using multi-channel audio feature with convolutional neural networks," *Proceedings of the Detection and Classification of Acoustic Scenes and Events*, 2020.

[184] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," *arXiv preprint arXiv:1711.10282*, 2017.

[185] A. Kumar, M. Khadkevich, and C. Fügen, "Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 326–330, IEEE, 2018.

[186] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011.

[187] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[188] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[189] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015.

[190] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *NIPS*, 2017.

[191] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[192] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with s-shaped rectified linear activation units," in *AAAI*, 2016.

[193] F. Agostinelli, M. D. Hoffman, P. J. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *CoRR*, vol. abs/1412.6830, 2014.

[194] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017.

[195] Q. Cheng, H. Li, Q. Wu, L. Ma, and N. N. King, "Parametric deformable exponential linear units for deep neural networks," *Neural Networks*, 2020.

[196] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[197] Y. Zhou, D. Li, S. Huo, and S.-Y. Kung, "Soft-root-sign activation function," *arXiv preprint arXiv:2003.00547*, 2020.

[198] G. Maguolo, L. Nanni, and S. Ghidoni, "Ensemble of convolutional neural networks trained with different activation functions," *Expert Systems with Applications*, vol. 166, p. 114048, 2021.

[199] L. Nanni, A. Lumini, S. Ghidoni, and G. Maguolo, "Comparisons among different stochastic selection of activation layers for convolutional neural networks for healthcare," *arXiv preprint arXiv:2011.11834*, 2020.

[200] S. Moccia, E. D. Momi, M. Guarnaschelli, M. Savazzi, A. Laborai, L. Guastini, G. Peretti, and L. S. Mattos, "Confident texture-based laryngeal tissue classification for early stage diagnosis support.," *Journal of medical imaging*, vol. 4 3, p. 034502, 2017.

[201] J. N. Kather, C. A. Weis, F. Bianconi, S. M. Melchers, L. R. Schad, T. Gaiser, A. Marx, and F. G. Zöllner, "Multi-class texture analysis in colorectal cancer histology," in *Scientific reports*, 2016.

[202] K. Dimitropoulos, P. Barmpoutis, C. Zioga, A. I. Kamas, K. Patsiaoura, and N. Grammalidis, "Grading of invasive breast carcinoma through grassmannian vlad encoding," in *PloS one*, 2017.

[203] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[204] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

[205] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[206] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[207] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[208] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[209] S. R. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, and B. B. Chaudhuri, "diffgrad: An optimization method for convolutional neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4500–4511, 2019.

[210] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, *et al.*, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," *arXiv preprint arXiv:2010.05627*, 2020.

[211] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628*, 2017.

[212] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[213] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[214] L. Nanni, G. Maguolo, and A. Lumini, "Exploiting adam-like optimization algorithms to improve the performance of convolutional neural networks," *arXiv preprint arXiv:2103.14689*, 2021.

[215] M. V. Boland and R. F. Murphy, "A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells," *Bioinformatics*, vol. 17 12, pp. 1213–23, 2001.

[216] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*, pp. 850–865, Springer, 2016.

[217] L. Nanni, S. Brahnam, A. Lumini, and G. Maguolo, "Animal sound classification using dissimilarity spaces," *Applied Sciences*, vol. 10, no. 23, p. 8578, 2020.

[218] P. Arora, S. Varshney, *et al.*, "Analysis of k-means and k-medoids algorithm for big data," *Procedia Computer Science*, vol. 78, pp. 507–512, 2016.

[219] M. S. Biagio, M. Crocco, M. Cristani, S. Martelli, and V. Murino, "Heterogeneous auto-similarities of characteristics (hasc): Exploiting relational information for classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 809–816, 2013.

[220] G. Maguolo and L. Nanni, "A critic evaluation of methods for covid-19 automatic detection from x-ray images," *Information Fusion*, vol. 76, pp. 1–7, 2021.

[221] O. Gozes, M. Frid-Adar, H. Greenspan, P. D. Browning, H. Zhang, W. Ji, A. Bernheim, and E. Siegel, "Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection & patient monitoring using deep learning ct image analysis," *arXiv preprint arXiv:2003.05037*, 2020.

[222] R. M. Pereira, D. Bertolini, L. O. Teixeira, C. N. Silla Jr, and Y. M. Costa, "Covid-19 identification in chest x-ray images on flat and hierarchical classification scenarios," *arXiv preprint arXiv:2004.05835*, 2020.

[223] A. Narin, C. Kaya, and Z. Pamuk, "Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks," *arXiv preprint arXiv:2003.10849*, 2020.

[224] J. Zhang, Y. Xie, Y. Li, C. Shen, and Y. Xia, "Covid-19 screening on chest x-ray images using deep learning based anomaly detection," *arXiv preprint arXiv:2003.12338*, 2020.

[225] I. D. Apostolopoulos and T. A. Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, p. 1, 2020.

[226] T. Ai, Z. Yang, H. Hou, C. Zhan, C. Chen, W. Lv, Q. Tao, Z. Sun, and L. Xia, "Correlation of chest ct and rt-pcr testing in coronavirus disease 2019 (covid-19) in china: a report of 1014 cases," *Radiology*, p. 200642, 2020.

[227] J. P. Cohen, M. Hashir, R. Brooks, and H. Bertrand, "On the limits of cross-domain generalization in automated x-ray prediction," *arXiv preprint arXiv:2002.02497*, 2020.

[228] E. H. Pooch, P. L. Ballester, and R. C. Barros, "Can we trust deep learning models diagnosis? the impact of domain shift in chest radiograph classification," *arXiv preprint arXiv:1909.01940*, 2019.

[229] J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann, "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study," *PLoS medicine*, vol. 15, no. 11, p. e1002683, 2018.

[230] J. Jo and Y. Bengio, "Measuring the tendency of cnns to learn surface statistical regularities," *arXiv preprint arXiv:1711.11561*, 2017.

[231] E. Tartaglione, C. A. Barbano, C. Berzovini, M. Calandri, and M. Grangetto, "Unveiling covid-19 from chest x-ray with deep learning: a hurdles race with small data," *arXiv preprint arXiv:2004.05405*, 2020.

[232] S. Tabik, A. Gómez-Ríos, J. Martín-Rodríguez, I. Sevillano-García, M. Rey-Area, D. Charte, E. Guirado, J. Suárez, J. Luengo, M. Valero-González, *et al.*, "Covidgr dataset and covid-sdnet methodology for predicting covid-19 based on chest x-ray images," *arXiv preprint arXiv:2006.01409*, 2020.

[233] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[234] I. M. Baltruschat, H. Nickisch, M. Grass, T. Knopp, and A. Saalbach, "Comparison of deep learning approaches for multi-label chest x-ray classification," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[235] L. Yao, J. Prosky, B. Covington, and K. Lyman, "A strong baseline for domain adaptation and generalization in medical imaging," *arXiv preprint arXiv:1904.01638*, 2019.

[236] J. P. Cohen, P. Morrison, and L. Dao, "Covid-19 image data collection," *arXiv preprint arXiv:2003.11597*, 2020.

[237] L. Wang and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest radiography images," *arXiv preprint arXiv:2003.09871*, 2020.

[238] I. Castiglioni, D. Ippolito, M. Interlenghi, C. B. Monti, C. Salvatore, S. Schiaffino, A. Polidori, D. Gandola, C. Messa, and F. Sardanelli, "Artificial intelligence applied on chest x-ray can aid in the diagnosis of covid-19 infection: a first experience from lombardy, italy," *medRxiv year=2020, publisher=Cold Spring Harbor Laboratory Press*.

[239] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, *et al.*, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 590–597, 2019.

[240] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.