Technische Universität Berlin
Faculty IV – Electrical Engineering and Computer Science
Institute for Software Technology and Theoretical Computer Science

Master Thesis

Machine learning techniques for the analysis of affective components of sign language

Neha Pravin Deshpande
Master of Science
Matriculation-Nr. 0453026

Berlin, 10.12.2021

Supervised by Examiners: Prof. Dr.-Ing. Sebastian Möller (TU Berlin), Prof. Dr. Gesche Joost (UdK Berlin). Advisors: Dr. phil. Eleftherios Avramidis, Dr. Fabrizio Nunnari (DFKI GmbH)

Acknowledgements

This thesis was supported by the SocialWear project of Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), and funded by the German Ministry of Education and Research (BMBF).

Foremost, I would like to thank my thesis advisors, Dr. Phil. Eleftherios Avramidis and Dr. Fabrizio Nunnari (DFKI). Without their assistance and involvement in every step throughout the process, this thesis would have never been accomplished. I would also like to show gratitude to my examiners, Prof. Dr.-Ing. Sebastian Möller (TU Berlin) and Prof. Dr.Gesche Joost (UdK Berlin). I would also like to thank Ms. Yasmin Hillebrenner for the administrative help she provided throughout this thesis. Last but not least, I am grateful to my family and friends for providing the cooperation and encouragement needed throughout this process.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich diese Diplomarbeit/Dissertation selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht.

Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt worden und auch nicht veröffentlicht worden.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Berlin, 10.12.2021 Neha.D

Ort, Datum, Unterschrift

Zusammenfassung

In dieser Arbeit wurden verschiedene Techniken des maschinellen Lernens untersucht, um eine bessere Genauigkeit für ein Modell zur Erkennung von Gesichtsausdrücken zu erreichen, das anhand von Gebärdensprachdaten trainiert wurde. Verschiedene Techniken des maschinellen Lernens wie Feinabstimmung, Datenerweiterung, Klassenausgleich sowie Bildvorverarbeitung wurden eingesetzt, um eine bessere Genauigkeit bei der Erkennung der sechs grundlegenden Ekman-Emotionen 'fear', 'disgust', 'surprise', 'sadness', 'happiness', 'anger' und der neutralen Klasse zu erreichen. Die Modelle wurden mittels K-facher Kreuzvalidierung evaluiert, um genauere Aussagen zu erhalten. In dieser Arbeit wird auch ein Vergleich der oben genannten Techniken auf der Grundlage von zwei verschiedenen Architekturen, nämlich MobileNet und EfficientNet, vorgestellt. Es wird experimentell gezeigt, dass die Feinabstimmung eines vortrainierten Modells zusammen mit der Datenerweiterung durch horizontales Spiegeln von Bildern und der Anwendung von Bildnormalisierung dazu beiträgt, die beste Genauigkeit für den Gebärdensprachdatensatz sowohl für MobileNet- als auch für EfficientNet-Architekturen zu erzielen.

Abstract

In this thesis, several machine learning techniques were studied to reach a better accuracy for a facial expression recognition model trained on a sign language dataset. Various machine learning techniques such as fine-tuning, data augmentation, class balancing, as well as image preprocessing were used to reach a better accuracy for recognizing the 6 basic Ekman emotions of ’fear’, ’disgust’, ’surprise’, ’sadness’, ’happiness’, ’anger’ along with the ’neutral’ class. The models were evaluated using K-fold cross-validation to get a more accurate conclusion. This thesis also presents a comparison of the above-mentioned techniques based on two different architectures, namely MobileNet and EfficientNet. It is experimentally demonstrated that fine-tuning a pre-trained model along with data augmentation by horizontally flipping images, and applying image normalization, helps in providing the best accuracy on the sign language dataset for both MobileNet and EfficientNet architectures.

Contents

1 Introduction

Sign language is a visual language that relies on movements of hands, body, as well as facial muscles to convey information. Most work on sign language recognition focuses on the movement of hands, which is considered a manual feature that conveys most of the information. There are also some non-manual features such as facial expressions, head and body position, and movement, which are also known to convey substantial information. Mukushev et al. (2020) in their research on non-manual features that differentiate similar signs in the Kazakh-Russian Sign Language, obtained a higher model accuracy overall when manual features were combined with non-manual features proving that the non-manual component of the sign language helps in improving a sign's recognition accuracy.

Facial expressions in sign language convey various types of meanings on linguistic and emotional information. These facial expressions are used in combination with other meaningful movements (those of hands, head, etc). According to Elliott et al. (2013) some facial expressions convey the inner emotional states. On the other hand, Fridlund (1997) claims that inner emotional states cannot be exactly read out from facial expressions. Facial expressions also have culture-specific meanings in sign language and hence other markers are needed to convey the real meaning behind a sign or a facial expression. Hence facial expression recognition is widely studied with different techniques used to improve the performance of models based on deep convolutional neural networks (CNN). Training a CNN requires a large amount of data and a limited amount of facial expressions data is available specifically for the German sign language, making it difficult to train a facial expression recognition model from scratch. Therefore, this thesis uses fine-tuning of pre-trained models that have provided a state-of-the-art accuracy on the AffectNet dataset (Savchenko 2021). The pre-trained models used during the experiments follow a lightweight architecture which makes it easier to fine-tune and still provides high accuracy.

For this study, it was hypothesized that fine-tuning a pre-trained facial expression recognition model (trained on a very large image dataset) helps improve the prediction rate on a sign language dataset. The sign language dataset used was the FePh (Facial Expression Phoenix) dataset which consists of 3000 facial images extracted from the daily news and weather forecast of the public TV station PHOENIX (Alaghband et al. 2020). This dataset is annotated with six basic emotions of 'sad', 'surprise', 'fear', 'angry', 'disgust', and 'happy' along with the 'neutral' and 'none' labels. The chosen approach for this thesis includes using fine-tuning first to test the performance of the model on the sign language dataset. This was followed by using various machine learning techniques such as data augmentation, image normalization, and class balancing to improve the performance of the fine-tuned model. As a final step, to evaluate the performance of the best models obtained with the techniques mentioned above, k-fold cross-validation was also performed.

The rest part of the thesis is organized as follows. A brief survey of related literature is given in Section 2. Section 3 includes a detailed description of the methods used during the thesis. Section 4 contains details about the experiments conducted for facial expression recognition. Section 5 presents the results of these experiments followed by Section 6 which concludes the thesis.

2 Related Work

As discussed in the previous section, to tackle the complexity of Facial Expression Recognition (FER), several machine learning techniques have been used. Ko (2018) has provided a brief review of the research done concerning FER in the past few years. This includes both conventional FER approaches as well as deep-learning-based approaches and a comparison of all the techniques based on certain evaluation metrics. The conventional approaches described by the researchers involve the extraction of geometric features of the face and then calculating the distance and angles between the facial landmarks to form a feature vector that can be used to train a machine learning model. Conventional machine learning algorithms such as Support Vector Machines (SVM), AdaBoost, or Random Forest classifiers, are then employed to perform the classification of the extracted landmarks. On the other hand, deep-learning-based approaches such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) can perform feature extraction, classification as well as recognition tasks. Visualization techniques can be used to see how these models learned with the given FER datasets. The paper also describes the use of video datasets, for which CNN can be combined with long Short-Term Memory (LSTM), where LSTMs can be used for temporal features and CNN for spatial features in an individual frame. The authors also provide a comparison of all the techniques mentioned above based on their accuracies, wherein the deep-learning-based techniques scored a higher average accuracy compared to the conventional techniques. This comparison shows that using deep-learning-based techniques with some fine-tuning can provide a better accuracy however, they still have several limitations, including the need for large-scale datasets, massive computing power, and large amounts of memory, and are time-consuming for both the training and testing phases as highlighted by the authors.

While describing the differences between traditional computer vision and deep learning techniques, O'Mahony et al. (2019) provided a similar review emphasizing that deep learning cannot solve all computer vision problems and hence a hybrid approach has also been in use lately. In addition to the review presented above, this paper also suggests that to tackle the issue of the requirement of massive data for deep learning, data augmentation techniques can be used as a preprocessing task to increase training data. Similarly, to reduce the training time taken by neural networks, transfer learning can be used (Akhand et al. 2021). Hence, focusing more on the state-of-the-art deep-learning-based techniques can be extended to recognize facial expressions in sign-language users.

Research suggests that some emotions are difficult to recognize for humans from just facial expressions. Dodich et al. (2014) proved that among negative emotions, fear is the most difficult to recognize while anger was recognized by most participants in the study. Facial expressions are culture-specific and findings by Sauter et al. (2010) indicate that several primarily negative emotions can be recognized across cultures, while most positive emotions are communicated with culture-specific signals.

2.1 Summary of existing Deep-Learning-based models

Some state-of-the-art techniques involving deep-learning-based approaches that can be used for facial expression recognition are presented below.

A multi-task training of lightweight convolutional neural networks for classification and identification of facial attributes has been presented by Savchenko (2021). They present a simple training pipeline that also provides a state-of-the-art accuracy of lightweight neural networks in FER trained on images as well as videos. Specifically, the emotion recognition network in the model is trained on the AffectNet dataset, providing a state-of-the-art accuracy. The high performance of this model is the result of pre-training of facial feature extractor for face identification, which was done by a very large VGGFace2 (Gennaro et al. 2019) data-set. The model also provides an excellent speed and model size, and hence can have several other applications as well. The features extracted by this network can be used with more complex classifiers, and therefore can be explored for FER in the case of sign language.

Another framework as proposed by Meng et al. (2019) involves the Frame Attention Networks (FAN) for video-based facial expression recognition. FAN takes a facial video consisting of frames of facial images as its input and produces a fixed-dimension feature representation which can be then used for facial expression recognition. With extensive experiments on CK+ and AFEW 8.0 data-sets (both including seven emotion labels (Ekman 1999)), the authors demonstrate that the framework with only self-attention improves the performance significantly, and adding relation-attention improves the performance further. FAN can be further fine-tuned to fit the sign-language dataset, due to its high accuracy.

Along with deep-learning-based models, there are also many pre-trained models involving Support Vector Machine (SVM) as well as Local Binary Patterns (LBP) (Ravi et al. 2020). LBP is a method for extracting features and the SVM classifier is used for classifying the features extracted from LBP. This paper also provides a comparison between LBP and CNN and proves that CNN does provide higher accuracy with some datasets (CK+ and JAFFE datasets) than SVM.

2.2 Techniques to improve performance of a CNN

To improve the performance of a Convolutional Neural Network (CNN) model, Savchenko (2021) employed data augmentation techniques which include geometric transformations such as flipping the training images horizontally, as well as cropping them randomly to increase the training data. Shorten et al. (2019) presents a more detailed review on the use of data augmentation techniques to expand limited datasets to take advantage of the capabilities of big data. In most computer vision tasks involving image classification, flipping the images horizontally before training is sufficient and helps in improving the overall performance of the CNN (Savchenko 2021; Zheng et al. 2020). Apart from data augmentation, research has also shown that using the right data preprocessing techniques such as resizing, face detection, cropping, adding noise, data normalization, histogram equalization, etc. also helps in boosting the performance of a CNN trained for recognizing emotions from facial images (Pitaloka et al. 2017).

Research done on different CNN architectures proves that architectures such as EfficientNet (B0 to B7), MobileNet, ResNet, etc. help in reducing the calculations required making them more lightweight and faster in performance (Tan et al. 2019, 2021). Along with the architecture, using several optimizers instead of just one also improves the overall performance and generalization of a CNN model (Savchenko 2021; Taqi et al. 2018). Taqi et al. (2018) used four different optimizers for a TensorFlow-CNN: Adagrad, ProximalAdagrad, Adam, and RMSProp to achieve accurate classification. While Adam is a commonly used optimizer giving a high classification accuracy on its own, using other optimizers that generalize better than Adam, can help improve the performance of the Adam optimizer further. The author also mentions the use of RMSProp optimizer which gave an accuracy of 100% while Adam optimizer ended up giving 96% accuracy. Savchenko (2021) used the Sharpness Aware Minimization (SAM) and Stochastic Gradient Descent (SGD) optimizers for the last few epochs as they converge better, boosting the overall performance. Other important parameters that could help boost the performance of a CNN are: using appropriate learning rates, choice of the activation function, balancing the imbalanced classes, etc (Kandel et al. 2020).

3 Methods

This section describes the various methods used in the experiments.

3.1 Image preprocessing

Image preprocessing plays a vital role in achieving state-of-the-art results in a Convolutional Neural Network (CNN), as the raw data does not always produce good accuracy. There are several image preprocessing techniques, such as image normalization, standardization, and Zero Component Analysis (ZCA) (Pal et al. 2016). The improvement in accuracy of a CNN is dependent on the image preprocessing technique being used along with its network architecture. This thesis uses two image preprocessing techniques, namely, face cropping and image normalization, which are discussed in the next subsection.

3.1.1 Face crop

Cropping is a technique used in computer vision to extract the area of the image which is required for image recognition or classification tasks. In the case of Facial Expression Recognition (FER), faces are cropped from the image dataset to remove the unnecessary information from the images, and only keep the pixels that constitute the facial information. To crop faces from an image, Savchenko (2021) has proposed the use of a Multi-task Cascaded Convolutional Network (MTCNN), a framework used for face detection and alignment. MTCNN performs three tasks: face classification, bounding box regression, and facial landmark localization (Xiang et al. 2017).

From the computer vision algorithms available to detect, recognize or crop faces such as Dlib and OpenCV libraries, the OpenCV library is more productive and has better performance for face detection (Boyko et al. 2018). OpenCV helps to crop the face from an image in the following steps: converting BGR images to RGB, detecting and extracting the face mesh from images, extracting the face bounds, and then finally cropping the images (Emami et al. 2012). OpenCV uses the Haar Cascade, which is an object detection method used to locate an object of interest in images. A Haar-like feature considers neighboring rectangular regions, sums up the pixel intensities in each region, and calculates the difference between these sums, which helps to categorize the image into subsections (Soo 2014).

3.1.2 Image normalization

As mentioned earlier, providing raw data to a CNN does not always help in providing good accuracy. Studies have shown that for image classification as well as recognition tasks such as predicting disease using X-rays, facial expression recognition, etc., image normalization has helped in enhancing the performance of the CNN (Heidari et al. 2020; Koo et al. 2017; Savchenko 2021).

Image normalization is a technique where the mean along each of the features (dimensions of images) from the training sample is calculated and is subtracted from each of the images. This results in normalizing the brightness of the whole training set concerning each dimension as shown in the equation below (Pal et al. 2016):

$$X' = X - \mu \tag{3.1}$$

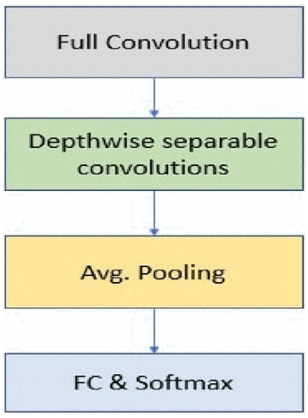

Fig. 3.1: MobileNet v1 baseline model (Sinha et al. 2019)

where X' is the normalized data, X represents the original data, and $\mu$is the mean vector across all features of X.

3.2 Convolutional Neural Network (CNN) Architectures

CNN's are deep learning models that play a significant role in Natural Language Processing (NLP), computer vision tasks such as image detection, recognition, etc (Albawi et al. 2017). Several CNN architectures have been developed to solve real-world problems including ResNet, MobileNet, DenseNet, EfficientNet, etc (Ioffe et al. 2015; Savchenko 2021; Sinha et al. 2019). As proposed by Savchenko (2021), for a fine-tuned facial expression recognition model, the EfficientNet architecture gave the highest accuracy compared to MobileNet. On the other hand, the MobileNet architecture is more lightweight and it works efficiently for a small number of parameters. The MobileNet and EfficientNet architectures are explained in the sections below.

3.2.1 The MobileNet architecture - MobileNet-v1

The MobileNet architecture (figure 3.1) uses depthwise separable convolutions followed by pointwise convolutions where each input channel is filtered separately as shown in figure 3.2. This results in a drastic reduction in model size and cost compared to standard convolutions. In comparison to other more efficient architectures, the accuracy obtained with MobileNet reduces as the number of parameters is increased in the model.

The MobileNet v1 architecture has 28 layers wherein each layer is followed by the batch normalization and Rectifier Linear Unit (ReLU) (Ioffe et al. 2015). The MobileNet v1 architecture starts with a regular 3×3 convolution, followed by 13 depthwise separable convolutional blocks and pointwise convolutions (Michele et al. 2019). The depthwise convolution in MobileNet is the channel-wise spatial convolution (Howard et al. 2017). Whereas the pointwise convolution is the 1×1 convolution which is used to change the dimension. This has been illustrated in the figure 3.2. These depthwise and pointwise convolutions result in a reduction in model size and computation cost by about 8 to 9 times as compared to the usage of standard convolutions (Sinha et al. 2019).

The MobileNet v1 architecture has been used for a variety of object detection and image recognition applications such as palm print recognition (Michele et al. 2019), handwriting character recognition (Ghosh et al. 2020), facial expression recognition (Savchenko 2021), etc.

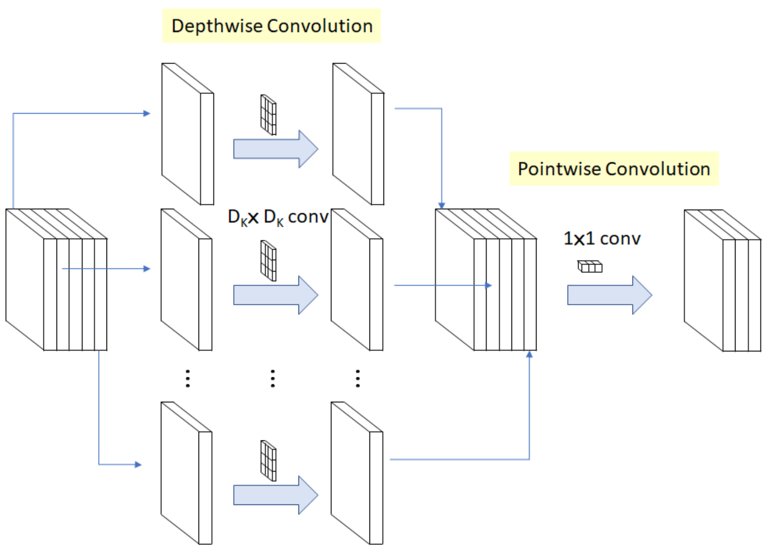

Fig. 3.2: Depthwise separable convolution operation followed by pointwise convolution (Howard et al. 2017)

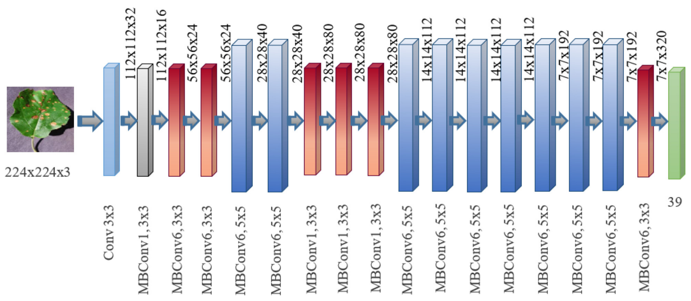

Fig. 3.3: Schematic representation of EfficientNet-B0 (Atila et al. 2021)

3.2.2 The EfficientNet architecture - EfficientNet-B0

The EfficientNet is another type of neural network architecture that consists of 8 models from EfficientNet-B0 to B7. The accuracy and the number of model parameters increase considerably with the model number. EfficientNet uses an activation function called Swish instead of Rectifier Linear Unit (ReLU) like the MobileNet architecture (Tan et al. 2019). The main building block for EfficientNet is the inverted bottleneck MBConv, which consists of a layer that first expands and then compresses the channel. (Sandler et al. 2018; Tan et al. 2019). This architecture has in-depth separable convolutions that reduce the calculation by almost k^2 factor compared to traditional layers, where k is the kernel size which denotes the width and height of the 2D convolution window (Sandler et al. 2018). A schematic representation of the EfficientNet-B0 is shown in figure 3.3. The EfficientNet architecture has been recently used for several applications such as plant leaf disease classification (Atila et al. 2021), automated diagnosis of COVID-19 (Marques et al. 2020), and other image classification tasks (Savchenko 2021; Tan et al. 2019), etc.

The EfficientNet architecture is more efficient than MobileNet and has provided state-of-the-art accuracy on several transfer learning datasets as it is easily scalable (Tan et al. 2019). Although when used for image classification problems, the EfficientNet architecture scaled up the image size leading to large memory consumption and slower training compared to MobileNet (Tan et al. 2019).

3.3 Training

3.3.1 Fine-tuning

Training a Convolutional Neural Network (CNN) from scratch requires a large amount of data. Instead, models pre-trained on bigger datasets can be fine-tuned to fit a low resource dataset. Fine-tuning is the process of initializing a pre-trained classification network and then training it further for a different task (Radenović et al. 2018). Particularly in the domain of Facial Expression Recognition (FER), fine-tuning has proved to have increased the classification accuracy (Akhand et al. 2021; Ngo et al. 2020). Fine-tuning helps in FER-related tasks, due to an imbalance in the facial emotion datasets available and the lack of training data, which can lead to overfitting and a less generalized model (Ngo et al. 2020).

As highlighted by Akhand et al. (2021), one of the motivations for using fine-tuning instead of fully training a model from scratch, is that the low-level basic features are common for most images and hence an already trained (pre-trained) model can be useful for classification by just fine-tuning the high-level features. Ngo et al. (2020) shows a comparison of several Facial Expression Recognition (FER) approaches including conventional techniques such as Gabor wavelets coefficients, Local binary pattern (LBP), Haar features, etc and Deep-learning based approaches such as CNN and transfer-learning based CNN. Out of these techniques, transfer learning-based CNN not only gives a highly accurate performance but also requires lesser human labor. The proposed FER technique as per (Akhand et al. 2021; Ngo et al. 2020; Savchenko 2021), is a pre-trained CNN modeled for image classification and fine-tuned by replacing the upper layers with the dense layer(s) to make it compatible with the fine-tuning dataset. Then the new dense layers are first tuned to the fine-tuning data, followed by training the whole CNN with this same data. After testing on several datasets such as AffectNet, KDEF, and JAFFE, fine-tuned CNN models provide a state-of-the-art accuracy (Savchenko 2021).

Akhand et al. (2021) has also mentioned the pipeline training strategy, which involves gradual fine-tuning of the model up layer-by-layer to achieve a high recognition accuracy. To implement a strategy like this one, the similarity of the pre-trained model and the target model should be considered.

3.3.2 Optimization in Neural Networks

The aim of a Convolution Neural Network is to learn from the given data by minimizing the loss. The loss function is reduced with the help of an optimization algorithm which is a numerical function performed on the model's parameters (Vani et al. 2019). An optimizer works towards reducing the loss incurred during the Neural Network's training process. Different optimizers are compared and analyzed by Vani et al. (2019) and Bera et al. (2020) in the context of deep learning and image classification.

As explained by Bera et al. (2020), the gradient descent algorithm is commonly used in neural networks for optimization as it minimizes the objective function by updating the parameters in the reverse direction of the gradient of the objective function. The cross-entropy value is a popular loss function that is equal to zero when the desired output and the predicted output are the same.

The following subsections explain the three optimizers used in this thesis for facial expression recognition including the Adam optimizer, Stochastic Gradient Descent (SGD), and the Sharpness Aware Minimization (SAM).

Adam optimizer

Adaptive Moment Estimation (Adam) is a method that computes discrete versatile learning rates for each parameter from the evaluation of the first and second moments of the gradients (Vani et al. 2019). It stores an exponentially decaying average of the past gradient (β_t) which represents the first moment (mean) and past squared gradient (γ_t) which represents the second moment (variance) (Bera et al. 2020). They are calculated as follows (Poojary et al. 2019):

$$\begin{cases} \beta_t \longleftarrow \rho_1 \beta_{t-1} + (1 - \rho_1)\, g_t \\ \gamma_t \longleftarrow \rho_2 \gamma_{t-1} + (1 - \rho_2)\, g_t^2 \end{cases} \tag{1}$$

As it only requires first-order gradients, it works with little memory requirement. Adam is also known to be robust and well-suited for a variety of machine learning problems (Ruder 2016, Kingma et al. 2014)

Stochastic Gradient Descent (SGD)

SGD uses a gradient descent algorithm but it takes data in samples while optimizing the CNN instead of considering the entire dataset at once (Poojary et al. 2019). Hence, reducing the number of factors and terms to be computed at each step. The SGD weight update rule is given in the equation below (Bera et al. 2020):

$$\theta_{t+1} = \theta_t - \eta d_t \tag{2}$$

where d_t represents the gradient of the objective function based on θ at time step t with η as the learning rate.

SGD is an optimizer that updates the model sequentially upon receiving new data and hence is suitable for big data analysis (Lei et al. 2020). Recently, there has been a significant amount of research pointing towards the algorithmic stability provided by SGD and has been proved to outperform other optimizers for fine-tuned CNN models (Lei et al. 2020, Li et al. 2018).

Sharpness Aware Minimization (SAM)

It is important for the neural network models to not only learn well on the given data but also generalize beyond the training data, which is done by reducing the training loss. Although a cross-entropy loss function is not sufficient to achieve generalization and hence choosing the right optimizer becomes essential (Foret et al. 2020). The author explains that sharpness Aware Minimization (SAM) is an optimization technique that seeks parameters that lie in neighborhoods having uniformly low loss leading to sub-optimal model quality. The authors also present empirical results showing that SAM improves the generalizability of the model across several datasets. SAM also provides robustness to noisy labels. SAM also helped achieve a better performance when applied on fine-tuned EfficientNet models pre-trained on ImageNet (Foret et al. 2020). Using SAM for optimizing the categorical cross-entropy loss for the last two epochs also provided a state-of-the-art accuracy on fine-tuned EfficientNet models pre-trained on ImageNet (Savchenko 2021).

From the experiments performed by Foret et al. (2020), SAM seeks out model parameters that are robust to perturbations suggesting SAM's potential to provide robustness to noise in the training set. Upon assessing the degree of robustness against label noise, SAM provided a high degree of robustness on par with the state-of-the-art procedures that specifically deal with noisy labels.

Foret et al. (2020) has also derived an algorithm for SAM using the Stochastic Gradient Descent (SGD) as the base optimizer, as it was seen to beat the performance of SGD on the CIFAR-10 dataset.

3.3.3 Data Augmentation

Data augmentation is another technique that helps compensate for the requirement of a large amount of data that is needed to train a deep learning model. Data augmentation relies on geometric transformations to increase the training data. Shorten et al. (2019) has explained several data augmentation techniques such as geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, Generative Adversarial Networks (GAN), neural style transfer, and meta-learning. Additionally, Porcu et al. (2020) also proposed a Facial Expression Recognition (FER) system using data augmentation techniques such as random rotation, horizontal and vertical flip, cropping, translation, and GAN. They recommend flipping the training images horizontally along with GAN as these techniques provide the most accurate improvement for FER.

A full-stage universal data augmentation framework is proposed by Zheng et al. (2020) as they explain that flipping data horizontally before feeding it to the CNN is not only safe but also one of the most common and effective data augmentation techniques. A horizontal flip of an image from the FePh dataset is shown in figure 3.5. The use of rotation and noise disturbance as data augmentation could have a large impact on the image structure if the images are small in size resulting in poor performance Zheng et al. 2020. TensorFlow allows flipping images horizontally with the help of Keras' preprocessing layers for data augmentation (Chollet 2016). A similar image transformation function is also available in PyTorch that enables easy image augmentation (Savchenko 2021).

Fig. 3.4: Original image from the FePh dataset

Fig. 3.5: Image flipped horizontally

3.3.4 Class Weights

The datasets available for facial expression recognition do not always consist of balanced classes as they have a different number of samples in each class. This can result in incorrect evaluation and a need for balancing these classes to achieve uniform results across classes. Johnson et al. (2019) has divided the techniques available to take care of the imbalance in classes into three categories: data-level methods, algorithm-level methods, and hybrid approaches. The data-level techniques involve under-sampling (wherein random samples from the majority class are discarded) and over-sampling (samples are duplicated in the minority class). On the other hand, the algorithm-based methods do not alter the training data, instead, the learning process is adjusted to compensate for the imbalance in the training data.

An algorithm-based technique used to balance the classes is called class weighting where different weights are used for every class depending on the number of training samples present in a class. The improvement in accuracy by using different class weights is demonstrated by Zhu et al. (2018) and Cardie et al. (1997) in case of medical applications. They proved that adding individual weights for each class instead of a single weight improved the recognition performance for minority classes while maintaining the same for the majority class. As explained by Johnson et al. (2019), class weights for each class can be calculated as follows:

$$cw = max_i|Ci|min_i|Ci|$$

Here, cw is the class weight for a minority class. Consider that the largest class in the dataset has 100 samples and the smallest class has 10 samples. If the class weight for the majority class is set to 1 then that for the minority class will be set to 10.

3.3.5 K-fold cross-validation

Cross-validation is a technique used to validate the generalizability of a model on a given dataset and is ideal to use cross-validation while working on small datasets (Yadav et al. 2016). cross-validation techniques such as k-fold cross-validation and hold-out can be used to train several models on different instances from the same dataset. Out of these two techniques, 0.1-3% more accurate results are obtained with the k-fold cross-validation technique (Yadav et al. 2016).

In K-fold cross-validation, the entire dataset is divided into k equal parts. Out of these k parts, 1 part is used as a test set, and the remaining $k - 1$ folds or parts are iteratively used for training models. Since the data being held out for testing is different in every iteration, the accuracy obtained with each model is also different. For example, for 5-fold cross-validation, the entire dataset is split into 5 folds, where 80% of the data is used for training while 20% is used for testing. Ultimately, 5 models are trained where each fold is used as a testing set once. After training and testing, the mean accuracy is considered as the final accuracy of the model (Yadav et al. 2016). The sklearn library of Python provides a Group K fold iterator that creates k-folds with non-overlapping groups (Bisong 2019).

According to Wong (2015) four factors affect the accuracy estimation while performing k-fold cross-validation namely: number of folds, number of instances in each fold, level of averaging, and repetition of k-fold cross-validation. A large value of k means there would not be enough instances of a class in each fold, although this will not be a problem in the case of large datasets. On the other hand, with a higher value of k, the computational cost increases (Fushiki 2011).

4 Experimental Setup

This chapter explains the experiments conducted to reach the best accuracy. As a starting point, the model by Savchenko, (2021) was used as it provided a state-of-the-art accuracy with the MobileNet and EfficientNet architectures. This model was considered as the baseline for predicting on the sign language dataset (FePh). This model was trained on a large AffectNet dataset (Mollahosseini et al. 2017) which contains more than 1,000,000 facial images from the internet. The model is first fine-tuned to the sign language dataset and then different machine learning techniques such as data augmentation, image preprocessing as well as class weight balancing, were used one after the other, to see which machine learning configuration gives the best accuracy on the sign language data. The above-mentioned machine learning techniques were used as they provided a state-of-the-art accuracy for the models presented in (Savchenko 2021). The following experiments were conducted one by one with each machine learning technique.
To train models with different configurations, it was necessary to use the same training and test sets throughout the experiments. Sections 4.1 to 4.3 explain the dataset used, preprocessing techniques used, as well as choosing the right pre-trained models to be used for further fine-tuning with the chosen dataset, respectively.

4.1 The sign language dataset

The dataset chosen for conducting the experiments was the Facial Expression Phoenix (FePh) dataset as introduced by Alaghband et al. (2020), which is an annotated sequenced facial expression dataset in the context of sign language, comprising over 3000 facial images extracted from the daily news and weather forecast of the public TV-station PHOENIX. The data was being annotated by the primary, secondary, and tertiary dyads of the six basic emotions of 'anger', 'disgust', 'fear', 'sad', 'happy', and 'surprise' along with the 'neutral' class. The 'none of the above class was also considered for images where no label could be assigned.

4.2 Data preprocessing

4.2.1 Removing the unknown data

As mentioned above, the FePh dataset was annotated with 7 labels, although the images that were not correctly recognized, were labeled as 'none of the above'. As this label did not fall under one of the 7 labels the pre-trained model was trained on, these images were removed from the dataset and were not used for further experiments. Along with this, the dataset also included many images with instances of multi-labeling where an image was labeled with more than one emotion. As this would result in a case of multi-label classification problem (Huang et al. 2019, Durand et al. 2019), such images were also removed from the final dataset.
After the removal of these images, the result obtained was a sequenced facial expression dataset with 2531 facial images annotated with 7 labels (6 emotions and the 'neutral' class).

Emotion	Data distribution
Anger	18.30%
Disgust	7.72%
Fear	12.43%
Happy	7.92%
Neutral	7.58%
Sad	14.36%
Surprise	31.85%

Tab. 4.1: Distribution of data across different emotion classes.

4.2.2 Face cropping

Another important technique for preprocessing images before feeding them to the CNN is cropping the face from the images to conform the FePh data to the dataset that the pre-trained models were trained on (AffectNet dataset). The images in the FePh dataset are not adequately cropped as they also include some parts of the upper body.

4.2.3 Splitting the train-test sets

The FePh dataset was first split into a training and test set with a split of 80% and 20% respectively. The images were split in such a way that the images belonging to the same video sequence were always kept together.

After this stage, the 80% split from the FePh dataset was further used to fine-tune the pre-trained models, and hence will be addressed as the fine-tuning dataset in the experiments explained below. The data distribution across the different emotion classes in the training set is as shown in table 4.1.

4.3 Pre-trained models for Facial Expression Recognition (FER)

There are many pre-trained models available for Facial Expression Recognition (FER), with almost all of them aiming to recognize the seven basic Ekman emotions. To get the state-of-the-art results, the technique presented in (Savchenko 2021) which provides a lightweight convolutional neural network for the recognition of facial emotions based on different architectures was chosen. With the models presented by the author, state-of-the-art accuracy was achieved on the AffectNet dataset (Mollahosseini et al. 2017), which includes almost 440k annotated images.

The pre-trained models further used for the experiments in this thesis were trained on the AffectNet dataset which in turn uses another pre-trained model trained on the very large VGGFace2 dataset (Gennaro et al. 2019). The two models that were further used in the experiments include: (1) a model based on the MobileNet architecture and (2) a model based on the EfficientNet architecture (Savchenko 2021). Both these models used techniques such as class balancing using class weights, data augmentation as well as image normalization.

The experiments conducted with both MobileNet and EfficientNet architectures are discussed in sections 4.4 to 4.6.

4.4 Experiments with the MobileNet architecture

For every experiment explained below, the model obtained after changing the hyperparameters was evaluated using the test set (20% FePh data split). The accuracy and sensitivity per class were measured for each experiment. To evaluate adequately on the test set, the same treatment was given to the test set as the fine-tuning set for a given experiment.

4.4.1 No-FT: Testing the pre-trained model on the FePh test set

The pre-trained model was presented by Savchenko (2021) and it was trained on the big AffectNet dataset with 440k annotated images with 7 basic Ekman emotions. This pre-trained model uses image normalization and horizontal flip as the data augmentation technique during the training process. To match the training set, the FePh test set was also treated with the same image normalization technique as explained before in Section 3.1.2. This experiment was performed to check how the existing pre-trained model performs on the sign language data set as the baseline for further experiments. From this experiment, the overall accuracy, as well as sensitivity per class, were recorded to compare with the results from the rest of the experiments.

4.4.2 FT: Simple Fine-tuning

After recording the results from Part A, the next approach was to check whether fine-tuning the pre-trained model with the FePh fine-tuning set helps in improving the accuracy and the sensitivity of the model on the test set. Fine-tuning was performed on the base model by Savchenko (2021).

Initially, a simple fine-tuning approach was used for the MobileNet architecture. In a Convolutional Neural Network (CNN), the last layer learns the high-level features, and hence the last few layers are sufficient for transfer learning (Tajbakhsh et al. 2016). The last layer of the pre-trained model was first removed and a new dense layer was added to the CNN and all the previous layers of the base net were frozen to train just the last layer. This last layer was then trained on the new dataset including images from the sign language (FePh) dataset for 3 epochs. Finally, all the previous frozen layers were unfrozen and the entire CNN was trained on the FePh data for 7 more epochs. The categorical cross-entropy loss was optimized by the Adam optimizer with a learning rate equal to 0.001.

4.4.3 FT-SGD: Fine-tuning with Stochastic Gradient Descent (SGD)

In this experiment, the approach proposed by Savchenko (2021) was followed wherein first the model was fine-tuned with Adam optimizer for 5 epochs and Stochastic Gradient Descent (SGD) was used for the last two epochs with the learning rate of 0.0001.

4.4.4 FT-SGD + CW: Class Weights for an imbalanced fine-tuning set

One of the reasons why, a CNN performs poorly, maybe the data distribution in the fine-tuning data. To tackle this imbalance across classes, the class weights for each class can be set separately (Zhu et al. 2018). Instead of assigning the same weight to every class, each class was assigned a different weight based on the data distribution in the fine-tuning dataset. This helps the minority classes that are insufficiently represented in the dataset. The data distribution across classes in the FePh fine-tuning dataset is shown in the table 4.1. The class weight parameter was used alone with fine-tuning to study the effects of balancing the class weights for each class in the FePh fine-tuning set.

Experiments	Configurations
No-FT	Base model tested on 20% FePh dataset
FT	Simple fine-tuning of base model with Adam optimizer
FT-SGD	Fine-tuning of base model with Adam and SGD optimizers
FT-SGD + CW	FT-SGD + Classes balanced with class weights
FT-SGD + HF	FT-SGD + Training dataset augmented with images flipped horizontally
FT-SGD + IP	FT-SGD + Images normalized before training
FT-SGD + IP + HF + CW	FT-SGD + Image normalization, horizontal flip and class weights
FT-SGD + IP + HF	FT-SGD + image normalization and horizontal flip

Tab. 4.2: Configurations used for the experiments involving the MobileNet architecture

4.4.5 FT-SGD + HF: Fine-tuning with Data Augmentation

For image recognition problems, the most popular data augmentation technique is to horizontally flip images before feeding them to the CNN. This technique was used during the training process to increase the fine-tuning data. This experiment used horizontal flip along with fine-tuning to study the effects of horizontal flip alone on the performance of the resulting model on the FePh test set.

4.4.6 FT-SGD + IP: Fine-tuning with Image Preprocessing

Image preprocessing was used along with fine-tuning where the fine-tuning data was normalized using the preprocessing function in Keras, which converts the images from RGB to BGR, then each color channel is zero-centered with respect to the ImageNet dataset (Ketkar 2017). This technique was also employed by Savchenko (2021), and hence was used independently with fine-tuning as in part B, to see the direct effects of horizontally flipping each image while training on the performance of the model.

4.4.7 Part G: Fine-tuning with combined methods

- FT-SGD + IP + HF + CW: This experiment was a direct replication of the one performed by Savchenko (2021), but by fine-tuning the pre-trained model with the FePh fine-tuning dataset. Since it had provided a state-of-the-art accuracy, this model was used to see if the combined effects of fine-tuning with data augmentation, image preprocessing, and class weights would improve the accuracy also with the FePh dataset compared to the previous experiments.

- FT-SGD + IP + HF: Fine-tuning with data augmentation and image preprocessing: The results obtained from the previous experiments proved that the two best approaches to include are data augmentation and image preprocessing as they provide better accuracy on the test set. Hence it was decided to train the model with this configuration.

All the experiments based on the MobileNet architecture are summarized in table 4.2 .

4.5 Experiments with the EfficientNet architecture

As discussed in Chapter 3, the EfficientNet architecture provides better accuracy on ImageNet and is considered a powerful tool in computer vision (Wang et al. 2021), (Savchenko 2021). Hence, those experiments that provided a higher accuracy from part A to part H were replicated for EfficientNet as well. Specifically, the EfficientNet-B0 architecture was used as the default input image size for the same is 224x224, which is

Experiments	Configurations
No-FT	Base model tested on 20% FePh dataset
FT + SAM + HF + CW	Base model fine-tuned with SAM optimizer + horizontal flip and class weights
FT + SAM + HF	Base model fine-tuned with SAM optimizer + horizontal flip
FT-SGD + SAM + HF	Base model fine-tuned with SAM and SGD optimizers + horizontal flip

Tab. 4.3: Configurations used for the experiments involving the EfficientNet architecture

the same as the size of the images in the dataset. The technique suggested by (Savchenko 2021) was used for recognizing facial emotions. The experiments performed with EfficientNet are listed below.

4.5.1 No-FT: Testing the pre-trained model on the FePh test set

This experiment is the same as explained in part A but was performed with the pre-trained model based on EfficientNet as proposed by (Savchenko 2021).

4.5.2 FT + SAM + HF + CW: Fine-tuning with Sharpness Aware Minimization, data augmentation as well as class weights

This experiment uses the replica of the pre-trained model provided by (Savchenko 2021), but with additional fine-tuning done using the FePh dataset. The procedure followed for fine-tuning is the same as explained in the Part B section of this chapter. However, this experiment uses the Sharpness Aware Minimization (SAM) as the optimizer, and initially, only the last layer is trained on the FePh fine-tuning dataset with a learning rate of 0.001 while freezing all layers in the base net. This last layer is trained for 3 epochs. Finally, all the layers are trained with the SAM optimizer with a learning rate of 0.0001 for 6 epochs as was proposed by Savchenko, (2021).
This experiment has also used horizontal flip as the data augmentation technique for increasing data while training.
As explained in the Part D section of this chapter, separate class weights can be set for each class due to a different data distribution in the training dataset. This technique is also implemented in this experiment as it helped achieve high accuracy for the model presented by (Savchenko 2021), which was trained on the AffectNet data.

4.5.3 FT + SAM + HF: Fine-tuning with SAM and data augmentation

To save time, the techniques that did not improve the accuracy in the case of the MobileNet architecture were dropped for EfficientNet as well. Hence, fine-tuning with Sharpness Aware Minimization (SAM) along with horizontal flip as the data augmentation technique was retained while the classes were left imbalanced as class weighting was removed from the model. The accuracy and the sensitivity per class were then recorded to compare the results with the experiment done in the previous subsection.

4.5.4 FT-SGD + SAM + HF: Fine-tuning with SAM and data augmentation and SGD

It was found that the best accuracy in the case of EfficientNet was obtained in the experiment conducted in part K. But the best model for the MobileNet architecture used Stochastic Gradient Descent as the optimizer while fine-tuning, and hence SGD was also tested in the case of EfficientNet.

4.6 5-Fold Cross-Validation

5-fold cross-validation was conducted by creating 5 sets of training and test sets from the FePh dataset by splitting the data by 80%-20% into training and test sets, respectively, 5 times. Each model was trained and evaluated on the FePh dataset. The accuracy and sensitivity per class were recorded for each model and the average accuracies and sensitivities along with the standard deviation for each class were calculated. cross-validation was performed for both MobileNet and EfficientNet architectures. Following are the experiments conducted with the MobileNet architecture:

- No-FT:
  As a starting point, the pre-trained model presented by (Savchenko 2021) was evaluated on the 5 test sets obtained after splitting the FePh data into 5 folds. The accuracy and sensitivity per class were calculated and then the average and standard deviation was calculated.

- cross-validation with configuration: FT-SGD + IP + HF + CW :
  This configuration as proposed by Savchenko, (2021) was initially used for cross-validation, since it provided a state-of-the-art accuracy with the AffectNet dataset.

- cross-validation with configuration FT-SGD + IP + HF :
  This configuration gave the highest accuracy on the FePh dataset and hence was chosen for cross-validation.

  cross-validation was also performed on models based on the EfficientNet architecture in the same way as mentioned above. The chosen configurations for cross-validation with EfficientNet were different and the ones which gave the best accuracies were chosen. The following points explain the experiments conducted with EfficientNet:

- No-FT:
  Similar to MobileNet, the pre-trained model or the base model presented by Savchenko, (2021), was considered as the starting point and was evaluated on the 5 test sets obtained after splitting the FePh data into 5 folds.

- cross-validation with configuration FT + SAM + HF + CW :
  This configuration was also chosen based on the one proposed by Savchenko, (2021) as it provided a high accuracy on the AffectNet dataset and the cross-validation was performed in the same way as explained above.

- cross-validation with configuration FT + SAM + HF:
  This configuration due to its high accuracy on the FePh dataset.

Table 4.4summarizes the configurations used during cross-validation.

The results of the experiments discussed above are presented in the next chapter. The experiments are denoted by the configuration name here.

Architecture	Configuration
MobileNet	No-FT
MobileNet	FT-SGD + IP + HF + CW
MobileNet	FT-SGD + IP + HF
EfficientNet	No-FT
EfficientNet	FT + SAM + HF + CW
EfficientNet	FT + SAM + HF

Tab. 4.4: Summary of the configurations used in 5-fold cross-validation experiments

5 Results

This section showcases the results obtained from experiments conducted with the MobileNet-v1, EfficientNet-B0 architecture, with both a single test set and 5-fold cross-validation. As evaluation metrics the model accuracy, sensitivity per class, and the average sensitivity was considered for every model. The result obtained from the experiment conducted without any fine-tuning involved is considered as the baseline result, and the results from the rest of the experiments are compared against this baseline. Finally, the cross-validation results were considered as the final results as it allows to compute mean and variance of the metrics across 5 folds, indicating the influence of random factors in the results. 5-fold cross-validation also helped in providing the average accuracies and sensitivity scores across five different models.

The following abbreviations are used for the techniques used during the experiments: FT: Fine-tuning, FT-SGD: Fine-tuning with the Stochastic Gradient Descent optimizer, IP: Image preprocessing, HF: Horizontal flip, CW: Class weights, SAM: Sharpness Aware Minimization.

5.1 Results with MobileNet-v1

From the results shown in table 5.1, fine-tuning improved the accuracy when the model was optimized with the Adam optimizer for the first few epochs and with the Stochastic Gradient Descent (SGD) optimizer for the last 3 epochs. It can be also seen that adding class weights to take care of the imbalanced classes reduced the overall accuracy, although when class weights were combined with image preprocessing and horizontal flip, it provided a better average sensitivity of 63.3%. The data augmentation technique of horizontally flipping the data alone did not provide any improvement in the accuracy of the fine-tuned model. Similarly, the use of image normalization as an image preprocessing technique did not help improve the accuracy. However, when data augmentation was combined with image normalization, the accuracy was increased to 67% providing the best accuracy across all the models trained.

5.2 Results with EfficientNet-B0

Table 5.2 shows that the best accuracy was provided by the configuration FT + SAM + HF. Similar to the MobileNet-v1 architecture, after removing class weighting from the base model as given by Savchenko (2021), the accuracy improved by 2.6%. The EfficientNet-B0 models took a long time to train (approximately one hour) due to the use of Sharpness Aware Minimization (SAM) (Foret et al. 2020) as the main optimizer compared to the MobileNet-v1 models which use the Adam optimizer for the most epochs and Stochastic Gradient Descent (SGD) for the last two epochs.

Configuration	Accuracy	Sensitivity per class (%)							Average
	(%)	anger	disgust	fear	happy	neutral	sadness	surprise	sensitivity (%)
No-FT	52.0	54.5	**74.2**	26.3	15.8	26.2	11.9	79.6	41.0
FT	65.4	**81.1**	38.7	45.6	63.1	35.7	50.8	77.8	56.1
FT-SGD	65.7	82.6	58.0	47.4	73.7	33.3	57.6	70.0	60.4
FT-SGD + CW	54.0	65.2	71.0	**56.1**	78.9	28.6	**61.0**	42.5	57.7
FT-SGD + HF	64.7	77.3	51.6	50.9	63.2	28.6	55.9	74.3	57.4
FT-SGD + IP	65.3	82.6	35.5	43.9	68.4	28.6	45.8	**80.2**	55.0
FT-SGD + IP + HF + CW	63.7	75.0	**74.2**	**56.1**	**84.2**	**38.1**	52.5	63.5	**63.3**
FT-SGD + IP + HF	**67.0**	81.8	61.3	40.4	68.4	33.3	50.8	79.6	59.3

Tab. 5.1: Accuracy, sensitivity per class and average sensitivity obtained for all the MobileNet-v1 configurations

Configuration	Accuracy	Sensitivity per class (%)							Average
	(%)	anger	disgust	fear	happy	neutral	sadness	surprise	sensitivity (%)
No-FT	53.5	50.8	67.7	**36.8**	47.4	47.6	18.6	73.1	49.0
FT + SAM + HF + CW	63.9	62.9	67.7	**36.8**	84.2	**76.2**	66.1	67.1	65.9
FT + SAM + HF	**66.5**	**68.2**	67.7	31.6	84.2	69.0	**67.8**	**73.7**	**66.1**
FT-SGD + SAM + HF	63.9	65.2	**71.0**	33.3	**89.5**	59.5	59.3	71.9	64.1

Tab. 5.2: Accuracy, sensitivity per class and average sensitivity obtained for all the EfficientNet-B0 configurations

5.3 Cross-validation results

Table 5.3 shows the results obtained after averaging the accuracies across 5 models trained while performing 5-fold cross-validation. The average accuracies and sensitivity for every class were recorded and it was found that the configurations that gave the best accuracies were FT-SGD + IP + HF for MobileNet-v1 with 62.4% accuracy on the 20% FePh test sets and FT + SAM + IP + HF for EfficientNet-B0 with an accuracy of 62.8% on the test sets. Since these accuracies are averaged over 5 models trained on different folds of training sets from the FePh dataset, these results can be considered more reliable (Yadav et al. 2016). Similar to the average, the standard deviation of accuracies and sensitivities of every class was also computed across the 5 models trained during cross-validation. The results for both MobileNet-v1 and EfficientNet-B0 architectures are displayed in the same table.

From table 5.3, it can be observed that fine-tuning with the full model outperforms the model with no fine-tuning. This confirms the hypothesis that fine-tuning the model on the FePh dataset helps to improve the performance of the model. It can be also seen that there is no significant difference between the best model trained with the MobileNet-v1 architecture and the EfficientNet-B0 architecture. No conclusion can be drawn regarding the comparison of the two architectures. From the best models obtained for both EfficientNet-B0 and MobileNet-v1, it can be concluded that class weighting hampered the performance of the models as it harms the accuracy obtained for particular classes. Nevertheless, the class weights did not have such a negative effect on the average accuracy of EfficientNet-B0, compared to their effect on MobileNet-v1. The best models obtained from 5-fold cross-validation in the case of both MobileNet-v1 (FT-SGD + IP + HF) and EfficientNet-B0 (FT + SAM + HF), also suggest that fear has the lowest recognition rate compared to all other classes (Table 5.3).

Architecture	Configuration	Average accuracy (std)	Average sensitivity per class (std)							Average sensitivity (std)
			anger	disgust	fear	happy	neutral	sadness	surprise	
MobileNet-v1	No-FT	44.1 (4.7)	44.4 (10.3)	**58.4 (2.4)**	**25.8 (12.4)**	47.3 (19.3)	20.8 (5.7)	17.2 (9.1)	63.8 (6.7)	39.7 (9.4)
	FT-SGD + IP + HF + CW	51.7 (7.4)	51.3 (21.4)	37.3 (13.8)	11.6 (5.5)	**56.6 (16.4)**	**75.3 (16.7)**	**60.4 (23.5)**	58.2 (12.0)	50.0 (15.6)
	FT-SGD + IP + HF	**62.4 (3.2)**	**73.7 (5.9)**	41.9 (14.1)	23.6 (9.6)	53.7 (12.2)	50.2 (22.3)	57.2 (16.1)	**82.1 (6.7)**	**54.6 (12.4)**
EfficientNet-B0	No-FT	45.7 (4.7)	42.3 (7.7)	55.6 (3.9)	30.1 (14.4)	53.3 (12.7)	54.6 (16.8)	19.5 (11.3)	59.7 (12.1)	45.0 (11.3)
	FT + SAM + HF + CW	62.2 (2.4)	**65.3 (9.3)**	**57.5 (12.6)**	**35.8 (13.3)**	72.3 (9.8)	**66.1 (10.0)**	**59.1 (16.8)**	68.5 (6.4)	**60.7 (11.2)**
	FT + SAM + HF	**62.8 (4.7)**	62.3 (8.6)	45.4 (18.8)	29.2 (16.6)	**82.7 (11.7)**	59 (17.6)	52.7 (22.2)	**79.2 (8.5)**	58.6 (14.9)

Tab. 5.3: The average accuracy, sensitivity per class, and average sensitivity (in %) with the standard deviation (std) obtained for all the MobileNet-v1 and EfficientNet-B0 configurations calculated after performing 5-fold cross-validation

6 Conclusion

From the experiments conducted, the hypothesis that fine-tuning with the full model improves the model performance compared to the one with no fine-tuning is confirmed. Although no significant difference was observed between the best configuration of MobileNet-v1 and EfficientNet-B0 models, the model based on the EfficientNet-B0 architecture still outperformed MobileNet-v1 by only 0.4% as seen from the cross-validation results. The training time for the EfficientNet models was higher than that of MobileNet models. The best configurations obtained with MobileNet took 45 minutes to train while the one obtained with EfficientNet took 1 hour due to the influence of Sharpness Aware Minimization (SAM) optimizer. From the cross-validation results, which are considered more reliable, it can be seen that the best models obtained with both MobileNet and EfficientNet, the 'fear' class has the lowest recognition rate out of the seven classes.

Every class seems to be profiting from different machine learning methods used during the experiments. The overall accuracy improved when image normalization was used in combination with augmenting training data with horizontally flipped images. Balancing classes with the help of class weights did not provide any significant improvement in the accuracy, despite the obvious lack of balance between the classes in the dataset.

Finally, the best models obtained from the experiments conducted were; for MobileNet v1 architecture: Fine-tuning with Stochastic Gradient Descent with training data normalized and augmented with horizontally flipped images, and for EfficientNet-B0 architecture: Fine-tuning with Sharpness Aware Minimization with training data augmented with horizontally flipped images.

Bibliography

Akhand, MAH, Shuvendu Roy, Nazmul Siddique, Md Abdus Samad Kamal, and Tetsuya Shimamura (2021): “Facial Emotion Recognition Using Transfer Learning in the Deep CNN”. In: *Electronics* 10.9, p. 1036.

Alaghband, Marie, Niloofar Yousefi, and Ivan Garibay (2020): “Facial Expression Phoenix (FePh): An Annotated Sequenced Dataset for Facial and Emotion-Specified Expressions in Sign Language”. In: *arXiv preprint arXiv:2003.08759*.

Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi (2017): “Understanding of a convolutional neural network”. In: *2017 International Conference on Engineering and Technology (ICET)*. Ieee, pp. 1–6.

Atila, Ümit, Murat Uçar, Kemal Akyol, and Emine Uçar (2021): “Plant leaf disease classification using EfficientNet deep learning model”. In: *Ecological Informatics* 61, p. 101182.

Bera, Somenath and Vimal K Shrivastava (2020): “Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification”. In: *International Journal of Remote Sensing* 41.7, pp. 2664–2683.

Bisong, Ekaba (2019): “More supervised machine learning techniques with scikit-learn”. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, pp. 287–308.

Boyko, Nataliya, Oleg Basystiuk, and Nataliya Shakhovska (2018): “Performance evaluation and comparison of software for face recognition, based on dlib and opencv library”. In: *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*. IEEE, pp. 478–482.

Cardie, Claire and Nicholas Howe (1997): “Improving minority class prediction using case-specific feature weights”. In:

Chollet, Francois (2016): “Building powerful image classification models using very little data”. In: *Keras Blog* 5.

Dodich, Alessandra, Chiara Cerami, Nicola Canessa, Chiara Crespi, Alessandra Marcone, Marta Arpone, Sabrina Realmuto, and Stefano F Cappa (2014): “Emotion recognition from facial expressions: a normative study of the Ekman 60-Faces Test in the Italian population”. In: *Neurological Sciences* 35.7, pp. 1015–1021.

Durand, Thibaut, Nazanin Mehrasa, and Greg Mori (2019): “Learning a deep convnet for multi-label classification with partial labels”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 647–657.

Ekman, Paul (1999): “Basic emotions”. In: *Handbook of cognition and emotion* 98.45-60, p. 16.

Elliott, Eeva Anita and Arthur M Jacobs (2013): “Facial expressions, emotions, and sign languages”. In: *Frontiers in psychology* 4, p. 115.

Emami, Shervin and Valentin Petrut Suciu (2012): “Facial recognition using OpenCV”. In: *Journal of Mobile, Embedded and Distributed Systems* 4.1, pp. 38–43.

Foret, Pierre, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur (2020): “Sharpness-aware minimization for efficiently improving generalization”. In: *arXiv preprint arXiv:2010.01412*.

Fridlund, Alan J (1997): “The new ethology of human facial expressions.” In:

Fushiki, Tadayoshi (2011): “Estimation of prediction error by using K-fold cross-validation”. In: *Statistics and Computing* 21.2, pp. 137–146.

Gennaro, Claudio and Claudio Vairo (2019): “Improving Multi-scale Face Recognition Using VGGFace2”. In: *New Trends in Image Analysis and Processing–ICIAP 2019: ICIAP International Workshops, BioFor, PatReCH, e-BADLE, DeepRetail, and Industrial Session, Trento, Italy, September 9–10, 2019, Revised Selected Papers*. Vol. 11808. Springer Nature, p. 21.

Ghosh, Tapotosh, Md Min-Ha-Zul Abedin, Shayer Mahmud Chowdhury, Zarin Tasnim, Tajbia Karim, SM Salim Reza, Sabrina Saika, and Mohammad Abu Yousuf (2020): "Bangla handwritten character recognition using MobileNet V1 architecture". In: *Bulletin of Electrical Engineering and Informatics* 9.6, pp. 2547–2554.

Heidari, Morteza, Seyedehnafiseh Mirniaharikandehei, Abolfazl Zargari Khuzani, Gopichandh Danala, Yuchen Qiu, and Bin Zheng (2020): "Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms". In: *International journal of medical informatics* 144, p. 104284.

Howard, Andrew G, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam (2017): "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861*.

Huang, Jun, Feng Qin, Xiao Zheng, Zekai Cheng, Zhixiang Yuan, Weigang Zhang, and Qingming Huang (2019): "Improving multi-label classification with missing labels by learning label-specific features". In: *Information Sciences* 492, pp. 124–146.

Ioffe, Sergey and Christian Szegedy (2015): "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR, pp. 448–456.

Johnson, Justin M and Taghi M Khoshgoftaar (2019): "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6.1, pp. 1–54.

Kandel, Ibrahem and Mauro Castelli (2020): "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset". In: *ICT express* 6.4, pp. 312–315.

Ketkar, Nikhil (2017): "Introduction to keras". In: *Deep learning with Python*. Springer, pp. 97–111.

Kingma, Diederik P and Jimmy Ba (2014): "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Ko, Byoung Chul (2018): "A brief review of facial emotion recognition based on visual information". In: *sensors* 18.2, p. 401.

Koo, Kyung-Mo and Eui-Young Cha (2017): "Image recognition performance enhancements using image normalization". In: *Human-centric Computing and Information Sciences* 7.1, pp. 1–11.

Lei, Yunwen and Yiming Ying (2020): "Fine-grained analysis of stability and generalization for stochastic gradient descent". In: *International Conference on Machine Learning*. PMLR, pp. 5809–5819.

Li, Yuanzhi and Yingyu Liang (2018): "Learning overparameterized neural networks via stochastic gradient descent on structured data". In: *arXiv preprint arXiv:1808.01204*.

Marques, Gonçalo, Deevyankar Agarwal, and Isabel de la Torre Dıez (2020): "Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network". In: *Applied soft computing* 96, p. 106691.

Meng, Debin, Xiaojiang Peng, Kai Wang, and Yu Qiao (2019): "Frame attention networks for facial expression recognition in videos". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 3866–3870.

Michele, Aurelia, Vincent Colin, and Diaz D Santika (2019): "Mobilenet convolutional neural networks and support vector machines for palmprint recognition". In: *Procedia Computer Science* 157, pp. 110–117.

Mollahosseini, Ali, Behzad Hasani, and Mohammad H Mahoor (2017): "Affectnet: A database for facial expression, valence, and arousal computing in the wild". In: *IEEE Transactions on Affective Computing* 10.1, pp. 18–31.

Mukushev, Medet, Arman Sabyrov, Alfarabi Imashev, Kenessary Koishybay, Vadim Kimmelman, and Anara Sandygulova (2020): "Evaluation of Manual and Non-manual Components for Sign Language Recognition". In: *Proceedings of The 12th Language Resources and Evaluation Conference*, pp. 6073–6078.

Ngo, Quan T and Seokhoon Yoon (2020): "Facial Expression Recognition Based on Weighted-Cluster Loss and Deep Transfer Learning Using a Highly Imbalanced Dataset". In: *Sensors* 20.9, p. 2639.

O'Mahony, Niall, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh (2019): "Deep learning vs. traditional computer vision". In: *Science and Information Conference*. Springer, pp. 128–144.

Pal, Kuntal Kumar and KS Sudeep (2016): “Preprocessing for image classification by convolutional neural networks”. In: *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, pp. 1778–1781.

Pitaloka, Diah Anggraeni, Ajeng Wulandari, T Basaruddin, and Dewi Yanti Liliana (2017): “Enhancing CNN with preprocessing stage in automatic emotion recognition”. In: *Procedia computer science* 116, pp. 523–529.

Poojary, Ramaprasad and Akul Pai (2019): “Comparative Study of Model Optimization Techniques in Fine-Tuned CNN Models”. In: *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. IEEE, pp. 1–4.

Porcu, Simone, Alessandro Floris, and Luigi Atzori (2020): “Evaluation of Data Augmentation Techniques for Facial Expression Recognition Systems”. In: *Electronics* 9.11, p. 1892.

Radenović, Filip, Giorgos Tolias, and Ondřej Chum (2018): “Fine-tuning CNN image retrieval with no human annotation”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.7, pp. 1655–1668.

Ravi, Rahul, SV Yadhukrishna, et al. (2020): “A face expression recognition using CNN & LBP”. In: *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, pp. 684–689.

Ruder, Sebastian (2016): “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747*.

Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen (2018): “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.

Sauter, Disa A, Frank Eisner, Paul Ekman, and Sophie K Scott (2010): “Cross-cultural recognition of basic emotions through nonverbal emotional vocalizations”. In: *Proceedings of the National Academy of Sciences* 107.6, pp. 2408–2412.

Savchenko, Andrey V (2021): “Facial expression and attributes recognition based on multi-task learning of lightweight neural networks”. In: *arXiv preprint arXiv:2103.17107*.

Shorten, Connor and Taghi M Khoshgoftaar (2019): “A survey on image data augmentation for deep learning”. In: *Journal of Big Data* 6.1, pp. 1–48.

Sinha, Debjyoti and Mohamed El-Sharkawy (2019): “Thin mobilenet: An enhanced mobilenet architecture”. In: *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, pp. 0280–0285.

Soo, Sander (2014): “Object detection using Haar-cascade Classifier”. In: *Institute of Computer Science, University of Tartu* 2.3, pp. 1–12.

Tajbakhsh, Nima, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang (2016): “Convolutional neural networks for medical image analysis: Full training or fine tuning?” In: *IEEE transactions on medical imaging* 35.5, pp. 1299–1312.

Tan, Mingxing and Quoc Le (2019): “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International Conference on Machine Learning*. PMLR, pp. 6105–6114.

Tan, Mingxing and Quoc V Le (2021): “Efficientnetv2: Smaller models and faster training”. In: *arXiv preprint arXiv:2104.00298*.

Taqi, Arwa Mohammed, Ahmed Awad, Fadwa Al-Azzo, and Mariofanna Milanova (2018): “The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance”. In: *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, pp. 140–145.

Vani, S and TV Madhusudhana Rao (2019): “An experimental approach towards the performance assessment of various optimizers on convolutional neural network”. In: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, pp. 331–336.

Wang, Kangrui and Xiaobing Yu (2021): “MobileNet and EfficientNet Demonstration on Google Landmark Recognition Dataset”. In: *International Core Journal of Engineering* 7.3, pp. 313–319.

Wong, Tzu-Tsung (2015): “Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation”. In: *Pattern Recognition* 48.9, pp. 2839–2846.

Xiang, Jia and Gengming Zhu (2017): “Joint face detection and facial expression recognition with MTCNN”. In: *2017 4th international conference on information science and control engineering (ICISCE)*. IEEE, pp. 424–427.

Yadav, Sanjay and Sanyam Shukla (2016): “Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification”. In: *2016 IEEE 6th International conference on advanced computing (IACC)*. IEEE, pp. 78–83.

Zheng, Qinghe, Mingqiang Yang, Xinyu Tian, Nan Jiang, and Deqiang Wang (2020): “A full stage data augmentation method in deep convolutional neural network for natural image classification”. In: *Discrete Dynamics in Nature and Society* 2020.

Zhu, Min, Jing Xia, Xiaoqing Jin, Molei Yan, Guolong Cai, Jing Yan, and Gangmin Ning (2018): “Class weights random forest algorithm for processing class imbalanced medical data”. In: *IEEE Access* 6, pp. 4641–4652.

List of Figures

List of Tables