Emotion Detection Based on VGGNet

Kangle Deng, Yiling Wang, Zhongyi Cao
Carnegie Mellon University
Pittsburgh, PA 15213
kangled@andrew.cmu.edu, yilinwan@andrew.cmu.edu, zhongyic@andrew.cmu.edu

Abstract

Facial expression recognition(FER) is one of the most important parts of emotion recognition process. Convolutional neural networks, particularly VGGNet, have shown great promise among all FER techniques due to its computational efficiency and outstanding performance. In this paper, we improve the test accuracy of emotion recognition on FER-2013 based on VGGNet model introduced by Khaireddin and Chen.[1] We experiment with various data pre-processing techniques, ensemble methods, and class-balanced loss. To the best of our knowledge, we achieve the best test accuracy of 72.4 percent by combining data pre-processing techniques, ensemble methods, and class-balanced loss.

1 Introduction

There are numerous physical symptoms of a person's mental state. Amongst them, facial expression is one of the most important parts of recognition of emotions. [2] Due to the empirical significance of emotion detection in sociable robotics, medical treatment, and many other human-computer systems, various studies have been conducted on automatic facial expression analysis.

In this paper, we aim to recognize each face based on the emotion shown in the facial expression into one of seven categories: angry, disgust, fear, happy, sad, surprise, and neutral. Our project goal is to improve the test accuracy of emotion recognition on FER-2013 based on VGGNet by employing different methods in data pre-processing, ensemble method, and changing loss.

2 Data

The dataset we choose to use is FER-2013[3]. It was introduced during the ICML 2013 Challenges in Representation Learning. The dataset consists of grayscale images of faces at a resolution of 48x48 pixels. It contains 28,709 training images, 3,589 validation images and 3,589 test images with seven categories: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral.

The Google image search API automatically collects FER-2013, which is an unrestricted and large-scale database. All images in the data set have been registered after eliminating frames with incorrect labels and modifying the cropped region.

It is suitable for this problem as the faces have been automatically registered such that they take up around the same amount of area and are more or less centerd. However, in unrestricted circumstances, non-facial variations like various illuminations and backdrops are quite prevalent [4]. Therefore, we should take these biases into account when evaluating the result.

Submitted to 10701 final project. Do not distribute.

Figure 1: Example images in FER-2013 dataset

3 Related Work

Convolutional Neural Networks(CNNs) [5] have shown promising potential in image processing since their introduction in the late 1990s. In a typical convolutional neural network, the structure of a convolutional layer, a pooling layer and a fully connected layer makes it efficient in manipulating static images. CNNs became increasingly more practical in feature extraction and image classification as processing power and the collecting of larger datasets increased in the 2010s[6].

CNNs have become a valuable tool for feature extraction, pattern recognition, and image processing, given all of the extensive research and improvements into them. When FER-2013 dataset was introduced during the ICML 2013 Challenges in Representation Learning, it was used as a benchmark for measuring model performance in the area of emotion recognition. Various models have shown impressive results, with classification accuracy ranging from 70.02 to 73.28 percent[7, 8]. And our facial expression recognition model improves on the VGGNet model presented by Khaireddin et.al[8], which achieves the best accuracy of 71.49 percent before fine-tuning.

4 Methods

Aside from the improvement listed below, we did not change the model architecture of VGGNet described by Khaireddin and Chen [1]. Figure 2 (adapted from Khaireddin and Chen [1]) shows the model architecture of VGGNet, which consists of 4 convolutional stages (each consists of two convolution blocks and a max pooling layer) and 3 fully connected (FC) feed-forward layers.

4.1 Data Pre-processing

In unrestricted scenarios, variations that are unrelated to face emotions, such as various backgrounds, illuminations, and head positions, are quite prevalent. Therefore, data pre-processing technique should be used to ensure the visual information supplied by the face is aligned and normalized.

4.1.1 Illumination Normalization

Images with low, non-zero contrast often include less information, which makes face expression detection difficult. Global Contrast Normalization(GCN) [9] divides each picture pixel value by standard deviation after removing it from the mean. Its goal is to eliminate pictures with different levels of contrast.

In our project, we employed Global Contrast Normalization on FER-2013 dataset. Figure 3 gives an example of image in FER-2013 after employing this technique.

The formula for GCN has the following notation and is defined as

- X: entire input image
- X': entire normalized output image

- $\bar{X}$: mean density of the entire image
- i, j, d: index of row, column, and image colour depth respectively
- λ: positive regularization parameter introduced to bias the estimate of standard deviation

$$X'_{i,j,d} = \frac{X_{i,j,d} - \bar{X}}{\max(\sqrt{\frac{1}{rc}\sum_{i=1}^{r}\sum_{j=1}^{c}\sum_{d=1}^{3}(X_{i,j,k} - \bar{X})^2 + \lambda}, \epsilon)}$$

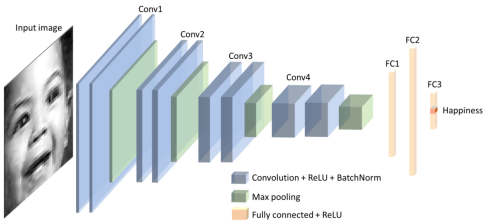

Figure 2: Architecture of VGGNet [1]

Figure 3: example image in FER-2013 before and after GCN

4.1.2 Facial Alignment

In data pre-processing, we also tried **Viola Jones face detector**[10], a well known method for recognizing near frontal faces to detect and remove background. Here is a brief explanation of this algorithm. The complete algorithm is more complicated, but the following 2 steps are the most essential[10]:

Integral Image The total sum of pixels above and to the left of x, y is contained in the integral picture at point x, y:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

where $ii(x, y)$ is the pixel value of (x, y) in the integral image.

Learning Classification Functions To choose a minimal number of features and train the classifier, a variation of adaBoost is utilized. The weak classifier aims to find the single rectangular feature that best distinguishes between negative and positive cases.

$$h_j(x) = \begin{cases} 1 & p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

where h_j is the j-th weak classifier in the weak learning algorithm, p_j is the sign symbol, f_j is the feature, and θ_j is the threshold.

4.2 Ensemble Method

Ensemble method aims to improve performance by combining multiple models. Past research indicates that ensemble method is effective in improving classification accuracy in emotion recognition. However, past research involving ensembling usually takes the form of taking the "majority" vote among models with different architecture, which naturally raises a question: Is it possible to apply ensemble methods without changing the architecture of the model?

AdaBoost-Based Ensemble A solution to this question is adaBoost. A description of the adaBoost-based ensemble is presented in Table 1. Aside from the standard adaBoost algorithm (**AB_Naive**), we introduced **AB_Cont** and **AB_Indpt**. The motivation for these two algorithms is to improve efficiency by initializing the models in later iterations to models trained in previous iterations, as deep neural networks often require lots of training before convergence.

VGG_EFN Aside from adaBoost, we also tried to ensemble models with different architecture. Proposed as an alternative to CNN, EfficientNets (EFN) has different architecture than VGGNet. We combine the output of VGG and EFN by creating output probability $\hat{p}_O = \hat{p}_{VGG} + \hat{p}_{EFN}$), where $\hat{p}_{VGG}$ and $\hat{p}_{EFN}$ are respectively the output probability of VGG and EFN. We then take the argmax of $\hat{p}_O$ as the predicted label.

Data & Parameters	
	We have in total N training samples and we train T VGGNets. Sample Weights $w \in \mathbb{R}^N$. Importance $\alpha \in \mathbb{R}^T$
Initialization	
	Initialize $w = (1/N, \cdots, 1/N)$ where $w[i] = 1/N \quad \forall i = 1, \cdots, N$ Initialize $\alpha = \vec{0}$ to be an empty vector. Set $t = 0$
Update Parameters	
	Once finish a iteration, use model h_t to generate prediction $\hat{y}$ on train set. Compute $\epsilon = \sum_{i=1}^{N} w_i^t \cdot \mathbb{I}(\hat{y} \neq y)$. Set $\alpha[t] \leftarrow \log((1-\epsilon)/\epsilon)$ Set $w_i^{(t)} \leftarrow \frac{w_i^{(t-1)}}{Z_t} \times \begin{cases} \exp(-1 \cdot \alpha[t]) & \hat{y}_i = y_i \\ \exp(+1 \cdot \alpha[t]) & \hat{y}_i \neq y_i \end{cases}$ where Z_t is the normalizing factor
Loss	
	We use weighted cross entropy (WCE) loss. Suppose for (x_i, y_i) the network gives $logits \in \mathbb{R}^7$. Then $\mathcal{L}_{WCE}(y_i, \hat{y}_i) = w_i^{(t-1)} \times \mathcal{L}(y_i, \hat{y}_i)$ where $\mathcal{L}(y_i, \hat{y}_i)$ is the cross entropy loss.
For $t = 0$	
	$t \leftarrow 1$. Train a VGGNet for 50 epochs. Save all checkpoints. **Update Parameters**
While $1 \leq t < T$**:**	
AB_Naive	Set $t \leftarrow t + 1$. Re-train 50 epochs from scratch under $w^{(t-1)}$. **Update Parameters** after finish training. Save all checkpoints
AB_Indpt	Set $t \leftarrow t + 1$. Load the last checkpoint from $t = 1$ and continue to train 25 epochs under $w^{(t-1)}$. **Update Parameters** after finish training. Save all checkpoints
AB_Cont	Set $t \leftarrow t + 1$. Load the last checkpoint from last (i.e., $(t-1)^{th}$) iteration. Continue to train 25 epochs under $w^{(t-1)}$. **Update Parameters** after finish training. Save all checkpoints
Prediction:	

	Now we have model $h^1, \cdots h^T$. $\hat{y}_i$ for sample (x_i, y_i) is calculated as follows:
	`logits = torch.zeros(7)  ## 7 is the number of labels`
	`for t in range(T):`
	`  logit_t = predict(x_i,model=h_t) #get logits from h_t`
	`  logits += alpha[t] * Softmax(logit_t, dim=1)`
	`pred_label = torch.argmax(logits, dim=1) ## predicted label`

Table 1: Algorithms for adaBoost-based ensemble methods. Different highlighting describes 3 variations of adaBoost. We only do the corresponding steps given the method.

4.3 Class-Balanced Loss

Class Imbalance Problem is quite common in machine learning: There are one or more classes (majority classes) that are more frequently occurring than the other classes (minority classes). This might be caused by different difficulty levels of collecting such samples or the underlying imbalanced distribution. Especially, Fig. 4 shows the number of samples in each class of FER2013. We find that

Disgust samples are significantly fewer than other classes while *Happy* samples are clearly more than others. However, most of the machine learning algorithms are based on the inherent assumption that the data is balanced, i.e., the data is equally distributed among all of its classes. This may lead to biased learned models with sub-optimal performance.

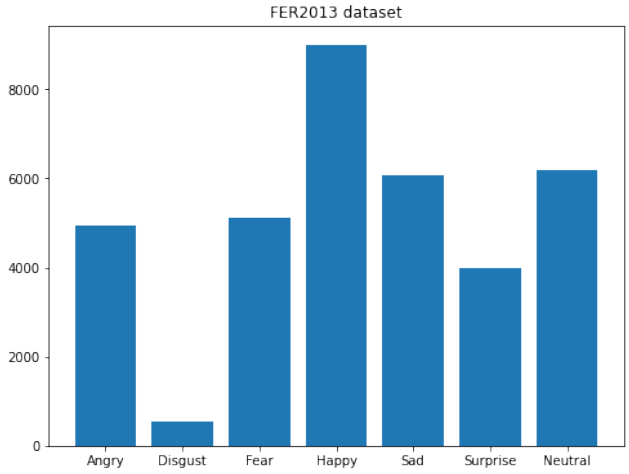

Figure 4: **# of samples in each class of FER2013:** We find that *Disgust* samples are significantly fewer than other classes while *Happy* samples are clearly more than others.

Class Weighting [11] is a modification for training losses to take into account the skewed distribution of the classes. This can be achieved by giving different weights to both the majority and minority classes. The difference in weights will influence the classification of the classes during the training phase. The whole purpose is to penalize the mis-classification made by the minority class by setting a higher class weight and at the same time reducing weight for the majority class. Specifically, for our facial expression classification problem, we typically use the following cross entropy loss to train:

$$l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\exp(\sum_{i=1}^{C} x_{n,i})} y_{n,c},$$

where x is the input, y is the target, and w_c is the weight for each class. Note $w_c = 1$ for regular cross entropy loss, which is used in the original VGGNet.

There are different weighting schemes that can be used to compute this class weight w_c. We tried three different weighting schemes: Inverse of Number of Samples (**INS**), Inverse of Square Root of Number of Samples (**ISNS**) and Effective Number of Samples (**ENS**) [11].

Inverse of Number of Samples (INS) is the most straightforward one that re-balances the loss based on the inverse of frequency for the class they belong to. We also normalize the weights over different classes to correct the scale of loss. This strategy is quite aggressive as it will lead to huge differences among weights.

$$w_c = \frac{1}{\text{\# of Samples in Class } c}.$$

Inverse of Square Root of Number of Samples (ISNS) is a more gentle re-weighting strategy that weights the samples as the inverse of the Square Root of frequency for the class they belong to.

$$w_c = \frac{1}{\sqrt{\text{\# of Samples in Class } c}}.$$

Effective Number of Samples (ENS)[11] is a general strategy based on the effective numbers. Training on an imbalanced dataset will make the model biased toward majority classes. However, inverse class frequency might be too aggressive, as it is putting too much emphasis on the minority class. We may want a parameterized re-weighting strategy that can be controlled between no-reweighting and INS. ENS weighting is given by the following formula:

$$w_c = \frac{1-\beta}{1-\beta^{n_c}},$$

where β is a hyper-parameter that controls the aggressiveness of the re-weighting. It is straightforward to see when $\beta = 0$, the strategy degenerates to no-reweighting (least aggressive). Furthermore, when $\beta \to 1$, the strategy approaches INS (most aggressive), as

$$\lim_{\beta \to 1} \frac{1-\beta}{1-\beta^{n_c}} = \frac{1}{n_c}.$$

5 Results

As we ran over 20 experiments., we present tables instead of loss progression graphs. Yet additional information is available in Appendix. Restricted by computation power, we did not include SE for most results as each experiment takes 5 to 10 hours to run on a 16 GB GPU, which is highly costly.

5.1 Experiment

We first tried to replicate the results by Khaireddin and Chen [1], who claimed the test accuracy of VGGNet trained over 300 epochs is 73.0%. However, under the same setting, we re-ran the experiment, but the test accuracy is only 71.4%. Thus, we will use 71.4% as the baseline measure.

We then run two stages of experiments. In the first stage, we independently explored the use of image pre-processing and class-balanced (CB) loss. In the second stage, we run 4 groups of experiments. The experiments are respectively characterized by the use of the original setup; pre-processing techniques; class-balanced loss; and both pre-processing techniques and CB loss. For each experiment, we examined the performance of different ensemble methods under the given settings. For all experiments, we used 0.01 as the learning rate and the SGD optimizer with Nesterov momentum under momentum of 0.9 and weight decay of 0.001. We also used Reduce Learning Rate on Plateau (RLRP) scheduler for all experiments. For experiments involving class-balance loss, we use $\beta = 0.999$, which has the best performance on the original VGGNet among other betas.

5.2 First Stage Results

Ablations on Re-weighting Strategy We run some experiments to determine the best strategy in our task. From Table 2, we can see that ISNS and ENS with a suitable β are able to outperform the baseline, which indicates the effectiveness of class weights. However, INS and ENS with larger β is too aggressive while ENS with smaller beta is too gentle. Therefore, we decide to use ENS ($\beta = 0.999$) in our final model.

Re-weighting Strategy	Test Accuracy
None	71.47%
INS	71.38%
ISNS	72.11%
ENS ($\beta = 0.9999$)	71.75%
ENS ($\beta = 0.999$)	**72.22%**
ENS ($\beta = 0.99$)	71.80%
ENS ($\beta = 0.9$)	71.66%

Table 2: Test accuracy for different re-weight strategies with different β

Image pre-processing By implementing and applying GCN to FER-2013 dataset, the test accuracy increase from 71.4% to 72.0%. This is consistent with our expectation that images with low, non-zero contrast often include less information and normalizing the data set can make face expression detection easier. The Viola-Jones algorithm, however, does not lead to improvement in our model, which is unexpected. Therefore, in the following experiment, we will refer pre-processing to GCN.

5.3 Second Stage Results

Table 3 presents the summary statistics of model performance under different settings. We see that without using any pre-processing techniques or the CB loss, the test accuracy of **VGG_EFN** and **AB_Indpt** has already exceeded the baseline VGGNet by 0.6%. Although the improvement is not very huge, it is already better than finetuning the model, as described by Khaireddin and Chen [1]

Among all the models we created, **VGG_EFN** with pre-processing and the original VGGNet with CB loss and pre-processing performs the best. The test accuracies of these two models are both 72.4%, which is 1% higher than the baseline. In addition, pre-processing seems to be complementary for **AB_Indpt**, while the CB loss seems to be helpful for **AB_Cont**. Regardless of the setting, **VGG_EFN** always out-perform the baseline model.

In Table 3, we see that ensemble-based methods can achieve similar results as VGGNet with much less training. As we prefer more computationally-light models, **VGG_EFN** with pre-processing should be considered as our best model. We re-ran this experiment 5 times. The mean test accuracy is 72.4% and the standard error is 0.1%, which suggests that our improvement is statistically significant.

Model	# of Epochs	Test Accuracy	TA w/Pre	TA w/CBL	TA w/CBL & Pre
VGGNet	300	71.4%	**72.0%**	**72.2%**	**72.4%**
VGG_EFN	$50 + 30$	**72.0%**	**72.4%**	**71.8%**	**71.9%**
AB-Naive	$50 + 50 \times 2$	71.1%	**71.4%**	71.3%	69.3%
AB-Indpt	$50 + 25 \times 2$	**72.0%**	**72.1%**	71.2%	70.7%
AB-Cont	$50 + 25 \times 2$	71.0%	71.0%	**71.6%**	70.9%

Table 3: Test accuracy of different ensemble methods. The 3^{rd} to 5^{th} columns are respectively test accuracy for ensemble methods with pre-processing; with class-balanced (CB) loss; and with both pre-processing and CB loss. All experiments involving adaBoost-based methods are trained over 3 iterations. Performance over the baseline is bolded.

Figure 5 shows loss and error progression for VGGNet with pre-processing and CB loss. We do not show such plots for **VGG_EFN** as it involves multiple models. We see that the validation accuracy converges at around 100 epochs.

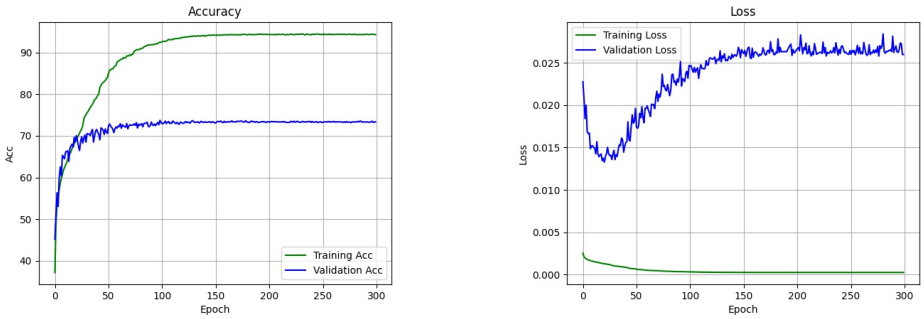

Figure 5: Loss and error progression for VGGNet with pre-processing and CB loss

6 Discussion and Analysis

6.1 Error Analysis

In Table 3, it appears unexpected that combining ensemble methods with CB loss performs worse than ensemble methods trained under regular cross entropy loss (except for **AB_Cont**). One possible explanation is that CB loss and adaBoost both addresses the class-imbalanced problem, thus applying them together overemphasized lower-resourced emotions and outliers, leading to a decrease in overall

performance. It is also unexpected that pre-processing and CB loss has opposite effect for **AB_Cont** and **AB_Indpt**, which is worth further research.

Table 4 presents an error analysis of our best-performing model, which is **VGG_EFN** with pre-processing. We see that the most common mistakes involve the misclassification between negative emotions (e.g., sad, angry, fear) and the mis-classification between neutral and other emotions.

True Label	Pred Label	Count
sad	neutral	103
fear	sad	74
sad	fear	68
sad	angry	54
neutral	sad	61
happy	neutral	49
fear	neutral	61

Table 4: Top 7 common most mistakes in **VGG_EFN** with pre-processing.

True Label	Pred Label	Count
fear	sad	139
neutral	sad	126
sad	fear	91
sad	angry	83
angry	sad	80
sad	neutral	76
surprise	fear	74

Table 5: Top 7 common mistakes in the 10^{th} iteration model of **AB_Indept** under original setting.

In addition, we noticed that making T very large for adaBoost-based methods does not improve the overall performance, as models in later iterations tend to perform poorly. We conducted similar error analysis for a model in the 10^{th} iterations of **AB_Indpt** without using CB loss or pre-processing. The results are presented in Table 5. We see that models in later iterations tend to have lower performance, and the misclassification between negative emotions seems to be more severe and dominant than in our best-performing model. It is surprising as "happy" is the most common label in the dataset, yet misclassification involving "happy" does not seem to be dominant in both Table 4 and Table 5.

6.2 Conclusion

The performance improvement by only employing data pre-processing technique suggests that variations in illumination, posture and face area affect model performance. It stresses the importance of normalized and aligned facial semantic information before training. Without changing the model architecture, adaBoost based algorithm can be used to improve model performance and to reduce required training. However, adaBoost need to be modified when used applied to deep neural models.

6.3 Limitations and Further Improvement

In section 6.1, we argued that for adaBoost-based methods, models in later iterations tend to perform poorly. This might be addressed by changing the update rule of w and α in the algorithm. Since adaBoost is primarily used for weak classifiers, it may need to be further modified when applied to deep neural models. Another possible solution is to increase training for models in later iterations, yet this is somewhat undesirable as we prefer lighter models. One particular ensemble method that we did not explore is bagging, which trains multiple VGGNet on bootstrapped samples. In error analysis, we noticed that the misclassification between neutral and other emotions is an major issue. To address this problem, further work may consider using contrastive learning objectives or generative adversarial networks (GAN) to better distinguish neutral and other emotions.

One limitation is that the performance of illumination normalization and background detection in reality might not be as good as in FER-2013, where the faces are approximately centered. In our project, we implemented Viola Jones (VJ) face detector to detect and remove the background. Yet using the VJ algorithm does not improve performance. It might because that FER-2013 have already been automatically registered such that they are approximately centered and occupy same amount of area. Though the VJ face detector is the sole process required for feature learning, face alignment utilizing localized landmark coordinates, which minimizes in-plane rotation variance, has also shown potential in improving FER performance. Therefore, future work could further explore face alignment utilizing localized landmark coordinates and more pose illumination techniques.

References

[1] Y. Khaireddin and Z. Chen, "Facial emotion recognition: State of the art performance on fer2013," 2021.

[2] P. Tarnowski, M. Kołodziej, A. Majkowski, and R. J. Rak, "Emotion recognition using facial expressions," *Procedia Computer Science*, Jun 2017.

[3] I. J. Goodfellow, D. Erhan, P. Luc Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, and et al., "Challenges in representation learning: A report on three machine learning contests," *Neural Networks*, vol. 64, p. 59–63, 2015.

[4] S. Li and W. Deng, "A deeper look at facial expression dataset bias," *IEEE Transactions on Affective Computing*, p. 1–1, 2020.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, p. 2278–2324, 1998.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, p. 84–90, 2017.

[7] S. Minaee, M. Minaei, and A. Abdolrashidi, "Deep-emotion: Facial expression recognition using attentional convolutional network," *Sensors*, vol. 21, no. 9, p. 3046, 2021.

[8] V. S. Amal, S. Suresh, and G. Deepa, "Real-time emotion recognition from facial expressions using convolutional neural network with fer2013 dataset," *Smart Innovation, Systems and Technologies*, p. 541–551, 2021.

[9] D. A. Pitaloka, A. Wulandari, T. Basaruddin, and D. Y. Liliana, "Enhancing cnn with preprocessing stage in automatic emotion recognition," *Procedia Computer Science*, vol. 116, p. 523–529, 2017.

[10] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.*

[11] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

7 Appendix

Architecture of VGGNet Figure 6 shows the model architecture of VGGNet. A VGGNet consists of 4 convolutional stages (each consists of two convolution blocks and a max pooling layer) and 3 fully connected (FC) layers. The loss used is cross entropy loss. For more details please refer to Khaireddin1 and Chen [1], especially Experiments section C.

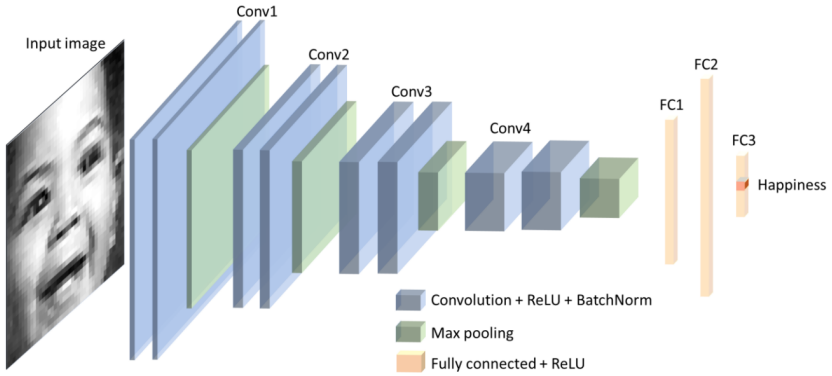

Figure 6: Architecture of VGGNet

Clarification on loss progression graph Due to the nature of ensemble methods, we cannot present a loss progression graph as it involves multiple models. We nevertheless provide some details. Some of the graphs are from experiments not presented in Results, but use the same methods with small differences (e.g., different number of epochs), For simplicity, all graphs presented below represent models without pre-processing or CB loss, as adding them result in similar graph.

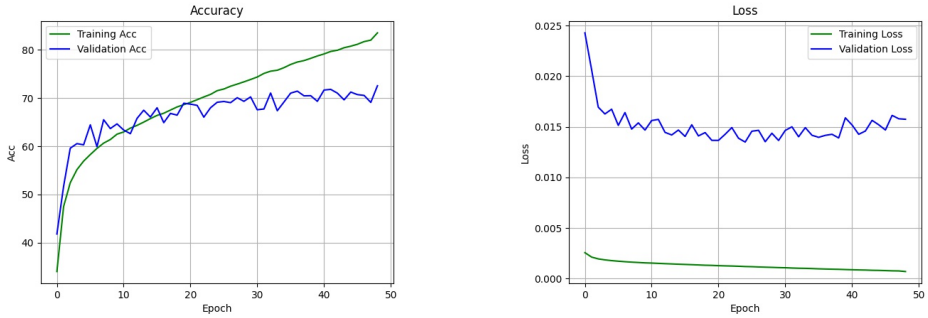

Figure 7: loss progression and validation accuracy progression for original VGGNet

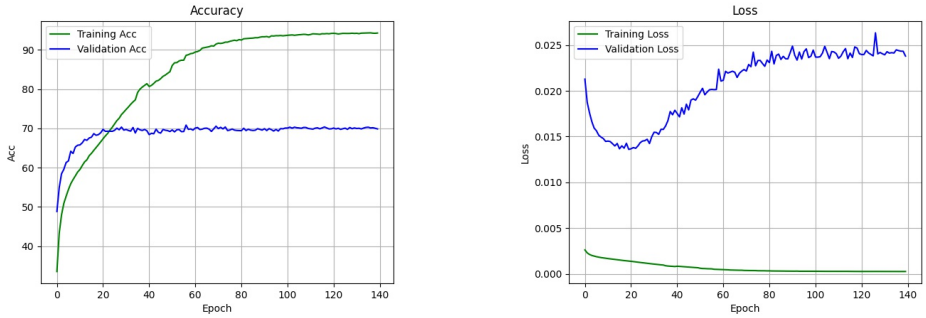

Figure 8: loss progression and validation accuracy progression for original EFN Model

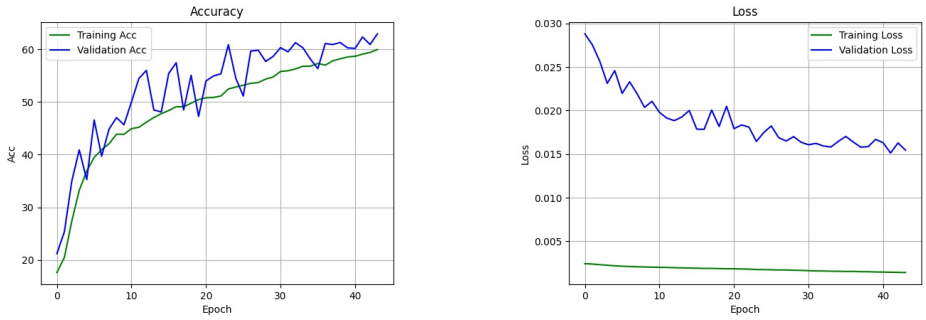

Figure 9: loss progression and validation accuracy progression for third model of **AB_Naive**

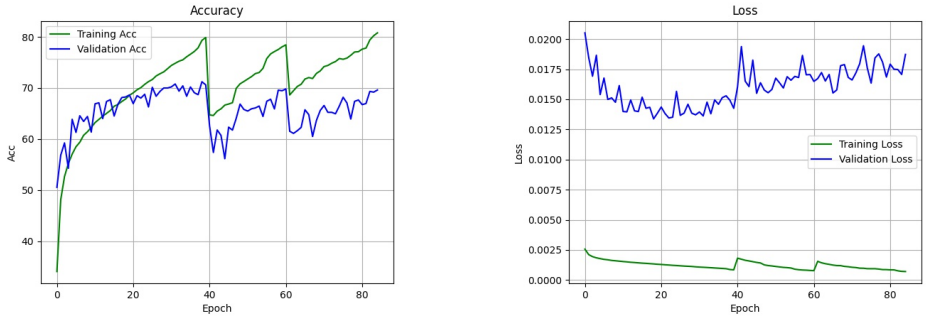

Figure 10: loss progression and validation accuracy progression for **AB_Cont** trained over 3 iterations

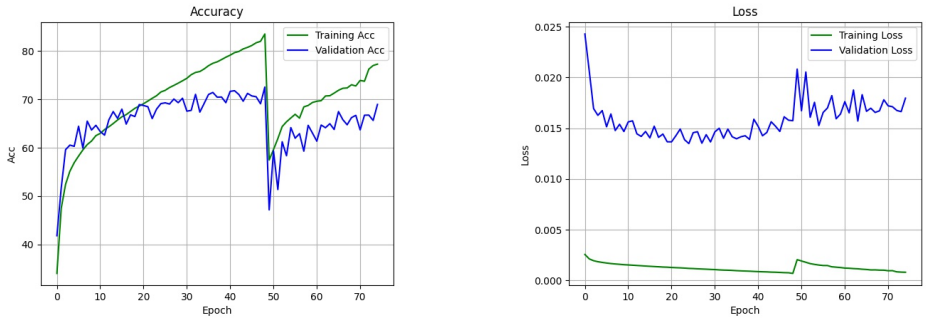

Figure 11: loss progression and validation accuracy progression for the first and third model of **AB_Indpt**. The third model begins with a sharp fall in accuracy around 50 epochs