Best Pretrained CNN: Fine-Tuning, Classifier Selection, and Ensemble Learning for Emotion Recognition

Anshu Singla
Chitkara University Institute of Engineering & Technology, Chitkara University,
Punjab, India
anshu.singla@chitkara.edu.in

Madhav Singh
Chitkara University Institute of Engineering & Technology, Chitkara University, (of Affiliation)
Punjab, India
madhavsinghabcde1@gmail.com

Abstract— **Facial emotion recognition (FER) is crucial for enhancing human-computer interactions, emotional intelligence systems, and psychological research. This paper presents a comprehensive analysis of fine-tuning popular convolutional neural networks (CNNs)—specifically ResNet-50, VGGNet-16, DenseNet-121, and EfficientNet-B0 on the AffectNet Dataset (8 emotion classes, 10,000 images). Comparison of the performances, strengths, and limitations of these architectures, providing proper step by step methods and analysis on Fine-tuning of Models was performed. Evaluated alternative classifiers (an Artificial Neural Network, Support Vector Machine, Random Forest, and XGBoost) using features extracted from each CNN's convolutional base. Finally, explored two ensemble strategies: (a) Feature-level ensemble – concatenating deep features from all models (after global pooling), optionally reducing dimensionality with PCA, and training a classifier; and (b) Prediction-level ensemble – combining the models' softmax outputs via learned meta-classifiers and through soft and hard voting. Experiments reveal that DenseNet-121 achieves superior accuracy due to its layers being interconnected to each other. SVM when used as a classifier, proves to give better accuracy than conventional ANNs used, if correct hyperparameters are chosen even in ensemble, SVM remains at top.**

Keywords— Facial Emotion Recognition, ResNet-16, VGGNet-16, DenseNet, EfficientNet, Transfer Learning, Hyper-Parameter Tuning

I. INTRODUCTION

Facial Emotion Recognition (FER) is a crucial task that aims to classify emotional state of human beings. FER can be beneficial for therapists, educators and caregivers to monitor the well-being of humans [1]. Recent advances in deep learning, especially Convolutional Neural Networks (CNNs) have dramatically improved the FER performance. Transfer learning [2, 3] via pretrained CNNs offers an effective solution: a CNN trained on a large dataset (e.g. ImageNet) can be fine-tuned on the FER task to leverage learned feature representations. This paper presents a comprehensive study of fine-tuning four widely used CNN architectures (VGG-16 [4], ResNet-50 [5], DenseNet-121 [6], EfficientNet-B0 [7] for FER on the subset of dataset of 8-class AffectNet, comparing their performance and optimization. The current study further addresses how best to perform classification using the CNN-extracted features, including traditional machine learning classifiers: Support Vector Machine (SVM) [8] and XG-Boost [9], Random forest classifier [10]) and ensemble methods [11]. Ensemble Learning [12] is motivated by the observation that different models may capture complementary facial features; by combining models, higher accuracy and robustness may be achieved. The authors explore both feature-level fusion (concatenating features from multiple CNNs) and prediction-level fusion (combining output probabilities) using various ensemble techniques. However, training CNNs from scratch requires large-labelled datasets. The AffectNet [14] database, for example, contains over 400,000 facial images labelled with eight emotions: neutral, happy, angry, sad, fear, surprise, disgust, contempt but using the full set is computationally intensive. In many scenarios, a smaller balanced subset (e.g. 10,000 images, ~1250 per class) is used to prototype models.

Prior Work: Researchers have applied transfer learning to FER with architectures like VGG-16, ResNet-50, DenseNet-121, etc., often achieving higher accuracy than training from scratch. For example, Fine-tuned ResNet-50, Xception, EfficientNet-B0, Inception, and DenseNet-121 on AffectNet and reported significant accuracy gains (5–10% absolute) over previous methods. But these models did not provide the layer by layer unfreezing method or its details, and how different models behave differently on specific blocks being unfrozen for weights training. The existing literature used SVM, Random Forests Classifier [13, 16] in place of fully connected ANN Layer. But, they did not provide proper comparison of various methods, to find out how each performs and what proves best for given conditions.

Ensemble strategies [17, 18] like voting or stacking/ bagging/boosting and have shown improvements in general image classification tasks, but their impact on FER with deep features is less explored. Here, Ensemble techniques were applied on both output level and CNN features level, to find out which proves best and by how much.

Contributions: (i) Four pretrained CNNs were finetuned on AffectNet and the methodology is documented for architecture modification and hyperparameter tuning, with detailed performance evaluation. **(ii)** Comparison of alternative classification strategies on CNN feature embeddings, hyperparameter tuning, comparison between different classifier heads. **(iii)** Design and testing of two ensemble pipelines (feature-level and prediction-level) to make the models overcome each others' weakness and check which ensemble methods prove beneficial and at minimal costs, along with providing the hyperparameters used and tested.

II. METHODOLOGY

FER is one of the challenging and delicate task as in day-to-day life as emotions are highly context-dependent ad require high-level precision. Fine-tuning a classifier for subtle emotions require a structured methodology to balance data preparation, model optimization, and further the evaluation. The detailed steps of methodology along with dataset selection and preparation are provided in subsequent subsections.

A. Dataset

AffectNet dataset [14] was utilized by selecting a balanced subset of approximately 10,000 images across eight distinct emotion categories (happiness, sadness, surprise, fear, disgust, anger, neutral, and contempt). AffectNet is an "in the wild" dataset, where natural images of humans in random conditions are chosen and annotated. Much of FER systems validate the model using AffectNet for the model accuracy, and for a long time, this dataset has remained as a valid benchmark for models' performance. The AffectNet dataset could potentially include a larger number of images, but doing so would come at a significant cost in terms of computation and time, besides, the only need is to showcase performance and efficiency, which suffice, by using 10,000 images, with images being equally distributed for all emotions.

B. Preprocessing

Images were resized to 224x224 pixels, normalized using mean and standard deviation. Since, original 96x96 AffectNet images perform poorly, due to feature extraction layers of these models, being trained on ImageNet dataset previously. All images were equally distributed amongst all classes, as these datasets have unequal number of images with respect to emotions, so that, model may not be trained or focussed heavily on some certain emotions and undermine the other emotions.

C. Model Architecture

Fine-tuning approach [7] utilized for prominent convolutional neural network (CNN) architectures, namely ResNet-50 (that favours gradual unfreezing), VGG-16 (that fine tunes dense layers) , DenseNet-121 (beneficial through regularization), and EfficientNet-B0 (enhances feature learning), to address the classification task. Models' convolution bases were imported, with a new ANN Top built that leverages influential pre-trained features and allows customization. To the convolution base, first a Global Average Pooling Layer was added followed by Batch Normalization Layer and then a Dropout (0.3), Dense (128, activation="ReLU"), Batch-Normalization, Dropout (0.3) and finally a Dense (8, activation="softmax") is applied i.e. fully connected layer with SoftMax activation function to classify 8 different type of emotions/classes. Global average pooling layer reduces the spatial dimensions on the other hand Batch normalization layer accelerates convergence.

D. Pretrained CNN Fine-Tuning Approach

The authors have employed a multi-phase fine-tuning strategy. The multi-phase fine tuning shifts the generic knowledge to task-specific knowledge and layered learning also reduces the chances of overfitting. Each phase can employ different loss functions, learning rates, or augmentation strategies to improve performance. The phase wise tuning strategy proven very useful in facial emotion recognition, where models must approach from generalized facial features to specific type of emotion. The following multi-phase tunings were utilized in the present study: Phase 1: Only the top dense layer was trained and rest all the base layers were frozen i.e. fine-tuned classifier head with epochs: 45 and learning rate (LR): 2e-4. Phase 2: Unfreeze last block in conv_base (CNN base) that means only the last block of base in trained and fine-tuned end-to-end using epochs: 30, and LR: 1e-5. Phase 3: Unfreeze second-last block in conv_base, fine-tuned end-to-end (epochs: 30, LR: 1e- 5). Phase 4: One by one unfreeze last blocks of conv_base and finetuned model, till reached a phase where performance of model worsened by unfreezing that particular block. Cross-entropy loss was adopted, Adam optimizer, and a batch size of 32. And used the best validation accuracy model weights by employing the use of Callbacks (Model Checkpoint). Model check points automatically save weights for which validation accuracy is high and loss is less which is very useful for hyperparameters tuning.

E. Alternate Classifiers for CNN Features

After loading the Pretrained and Fine-tuned models, the models were sliced after the Global Average Pooling Layer, but, also including the Batch Normalization Layer present right after it. This reduces additional steps of normalization and preprocessing while using different Algorithms/models for those Predictions.

Next, we passed those predictions to different Models, Random Forests (Bagging), XG-Boost (Boosting), SVM, to check their predictions. Various loops and Hyper Parameters were added to find out the best hyperparameters with maximum validation accuracy (val_accuracy). Tests were only for performed for DenseNet, since it showed maximum val_accuracy with just a top ANN layer and no convolution base finetuning. The results of this could be beneficial for understanding the impact of different types of classifiers.

F. Ensemble Methods

These were performed by two ways, first, employing ensemble at prediction level, second, at feature level, that is, right after convolution base and its following Global Average Pooling and Batch Normalization layers where different approaches, such as, Hard and Soft Voting, ANN network finetuning, SVG, Random Forests, XG-Boost and AdaBoost were used.

III. EXPERIMENTAL RESULTS AND ANALYSIS

Four models VGG-16, Resnet-50, DenseNet121, and EfficientNet-B0 along with SVM have been employed to fine tune the model. The experiments along with results have been explained in subsequent section.

A. Fine-Tuned CNN Performance

VGG-16 and Resnet-50 showed improvement in performances with each block of convolution base being unfrozen for training. But, in some cases, we could not go too far with that, since when approached to the last layer or second last layer being unfrozen, the performance started degrading, since the major feature extraction layers' weights started degrading for Resnet-50.

DenseNet-121 showed degradation in the last Block being unfrozen itself, the reason being the interconnection of all layers in DenseNet-121 network. The co-dependence of all layers working together in a DenseNet-121 could be reason for this phenomenon. This is the reason the authors unlocked all layers' weights, in the fine-tuning phase of convolution base (conv_base) , and it performed great results, better than all other models.

EfficientNet-B0 performed worse from starting. But, as its conv_base was unfrozen block by block, its performance started improving, but not too much in comparison with other models on the benchmark of validation accuracy i.e. val_accuracy.

For models, in each block unlocking phase, best accuracies (val_accuracy) at about 14-18 epochs were observed, which models' weights were picked for next block unfreezing and fine-tuning phase. Experimental results in Table 1 and Table 2 highlights DenseNet-121's performance above all other models, also it was reached just within few epochs, with no need of going through many conv_base blocks unfreezing phases one by one, so as to reach the maximum val_accuracy model at last, as if all blocks were unfrozen at once, models would have performed worse, DenseNet-121, ResNet-50, and VggNet-16 models also showed improvement in their performances with each phase, most importantly noting that, one should not unfreeze blocks to the very last, which in case, worsens the model performance by degrading key feature extraction weights learned by the model from large image-net datasets.

Table-1: Validation Accuracy Comparison for respective Block Finetuning for given Models

Models	Validation Accuracy				
	Only Top Trained	Last Block Unfrozen	Second Last Block Unfrozen	Third Last Block Unfrozen	All Blocks Unfrozen
ResNet-50	42	46	48	49	20
VggNet-16	45	58	59	57	59
DenseNet-121	48	15	-	-	**60**
EfficientNet-B0	13	15	25	30	-

** Highest validation accuracy is shown in bold*

Table-2: Validation Loss Comparison for respective Block Finetuning for given Models

Models	Validation Loss				
	Only Top Trained	Last Block Unfrozen	2nd Last Block Unfrozen	3rd Last Block Unfrozen	All Blocks Unfrozen
ResNet-50	1.46	1.57	1.95	2.04	3.5
VggNet-16	1.45	1.26	1.54	1.88	1.9
DenseNet-121	**1.24**	2.05	-	-	1.3
EfficientNet-B0	2.07	2.08	1.75	1.79	-

** Minimum Validation Loss is shown in bold*

B. *Alternative Classifier results on Deep Features*

SVM and Random Forests were fit on the CNN Features of the DenseNet-121 conv_base after Global Average Pooling and Batch Normalization. It was observed that SVM reached a slight up of val_accuracy compared to our conventionally used ANN classifier head. While ANN classifier head, gave a val_accuracy of 48% after 15 epochs, SVM in Table 3 gave 48.4%, using the radial basis kernel named 'rbf' kernel. It seems like a good choice at first, but, since SVM cannot be trained, like ANN would be after unfreezing the remaining CNN layers, thus, ANN classifier model val_accuracy would improve by many times, in second round of fine tuning. Also, it was observed that Random Forests in Table-4 gave lower validation accuracies, but it is just due to the decision trees weaker ability to deal with this kind of data, and this val_accuracy can be still impressive for Random Forests. Sometimes, it is discussed that 'Categorical Hinge' loss function can be used, if SVM like concept needs to be used and also, since, SVM cannot be fine-tuned with the model, just like ANN classifier back propagates. So, we tried Categorical Hinge to overcome the shortcomings of SVM, but the result was worse.

Table-3: Validation Accuracy comparison for SVM with Different hyper-parameters

S.No.	Hyper-parameters				Validation Accuracy (val-accuracy)
	Kernel	Class weight	Estimator (C)	Gamma	
1.	Rbf	Balanced	1.0	Scale	**48.4**
2.	Rbf	Balanced	1	0.01	47.8
3.	Poly	Balanced	1	Scale	47.8
4.	Rbf	Balanced	0.1	Scale	44.9

** Highest validation accuracy is shown in bold*

The main reason is that Categorical hinge loss penalizes the model when the score of correct class is low. The scores cannot be normalized by SVM as it does not employ softmax activation which further results in unstable gradients and poor convergence, when combined with deep learning-based frameworks. It can be observed in first round: 43% val_acc and 1.05 val_loss (validation loss). And on further training after unfreezing of weights, 23% val_acc and 1.31 val_loss

C. *Different Ensemble Models and their Performances*

1) *Prediction Level Ensemble*

As shown in Fig-1, model structure was used with varying Ensemble methods and comparison can be seen in Table-5. First, ANN [Dense(16, 'ReLU'), Dense(8, 'softmax'))] classifier head ensemble was created and analyzed, with the rest of the three models frozen along all their layers. The layers were trained first at learning rate(lr) of '2e-4' using 'Adam' as optimizer for 30 epochs, receiving a val_accuracy of 0.57 and val_loss 1.52. Then trained for 15 more epochs, at a learning rate of '1e-3', showing a val_accuracy of 0.5881. Which is just comparable to DenseNet trained. **Hard Vote** gave val_accuracy of 0.5909 while **Soft Vote** gave val_accuracy of 0.6169. And there was no hyperparameter tuning required for this.

As a result the cons is it does not try to improve or fine-tune, or that results do not even check y_train. It just does so without use of validation or train data. So, wrong models do not necessarily improve with time but may also spoil other models results as well. Random Forests ('n_estimators': 100, 'max_features': 'log2', 'max_depth': 10, 'min_samples_split': 10, 'min_samples_leaf': 1, 'val_log_loss': 1.856) ensembles achieved a validation accuracy of 0.615—an impressive result given their rapid convergence. However, tuning their hyperparameters can be

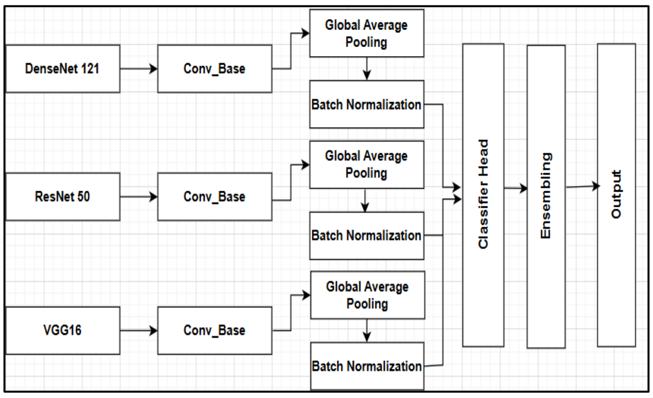

Fig-1: Methodology followed by Prediction Ensemble Model.

time consuming, and performance may degrade as the dataset dimensions grow. Nevertheless, Random Forests remained highly effective for this type of data and classification task. SVM (‘val_accuracy’: 0.6256, ‘C’: 10, ‘kernel’: ‘poly’, ‘gamma’: ‘scale’, ‘degree’: 2, ‘class_weight’: None) using One versus rest classifier, was used, and it performed even better than RF or other techniques, ‘poly’ and ‘rbf’ kernels deemed to help greatly in classifying these data spread in multi-dimensional space. Regularizer ‘scale’ deemed with specific degree for complex tasks, makes it highly optimized to be task specific. But, this is the very thing, that makes it extremely time consuming and slow to find the best hyperparameters given certain situations. So, it provides good results and can be as much as computationally expensive as well. XGBoost (‘n_estimators’: 100, ‘lr’: 0.1, ‘Max_Depth’: 5, ‘min_child_weight’: 3, ‘gamma’: 0.2, ‘subsample’: 0.7, ‘colsample_bytree’: 0.7, ‘reg_alpha’: 0, ‘reg_lambda’: 2): XG-Boost tried its best and reached a val_loss accuracy of 0.5969, but performed lower to Random Forest, since, It needs clean data to process, while noisy data can disturb its algorithms. While directly working Random Forests, it sets all individual decision trees that contains multiple nodes and collects final decisions by bagging. AdaBoost (val_accuracy: 60.0625, ‘n_estimators’: 150, ‘lr’: 0.1, ‘max_depth’: 4, ‘min_samples_split’: 10, ‘min_samples_leaf’: 4) repeatedly up weights the hardest examples, so if your meta features include noisy or borderline cases, it will focus on those and can overfit to noise or outliers. Its weak learners (often decision stumps) may be too simple to capture the subtilities in the concatenated CNN probabilities, so the boosting process amplifies errors rather than smoothing them out. As the Logistic Regression (‘val_accuracy’: 0.6169, ‘solver’: ‘liblinear’, ‘C’: 0.01) does not work on focusing heavily on certain parts, it too performed better than XG-Boost and AdaBoost. It was noticed mainly that, even-though XG-Boost and AdaBoost are very powerful training methods, but they still tend to perform lower to models are Random Forests Classifier and Logistic Regression, mostly because of the type of data handled. The act of improving the errors and hard examples and fixating those becomes the very reason for their lower val_accuracies. They might have performed better if the data were different. SVM used wrapped inside One versus rest classifier, due to its high optimizations and extreme flexibility performed the best, but finding the right hyperparameters for it, is just as difficult and both time and computationally expensive.

Table-4: Validation Accuracy Comparison for Prediction Ensemble Models

Model	Accuracy (%)	Model	Accuracy (%)
SVM	**62.56**	AdaBoost	60.63
Logistic Regression	61.69	XG-Boost	59.69
SoftVote	61.69	HardVote	59.09
Random Forests	61.5		

** Highest validation accuracy is shown in bold*

2) *Features Level Ensemble*

As shown in Fig-2, model structure was used with varying Ensemble methods and comparison can be seen in Table-6. Ensemble creation on the conv_base layers as well, would build a true ensemble model, one which takes multiple models’ inputs and not just preds, for predictions. Since, the conv_base features would be too many, and we need to combine those of three models to proceed, which would increase those features by three times. So, authors’ applied Global Average Pooling and Batch Normalization before taking those inputs. Additionally, after concatenating those values, we decreased dimensions by using PCA, to avoid noisy data and faster convergence.

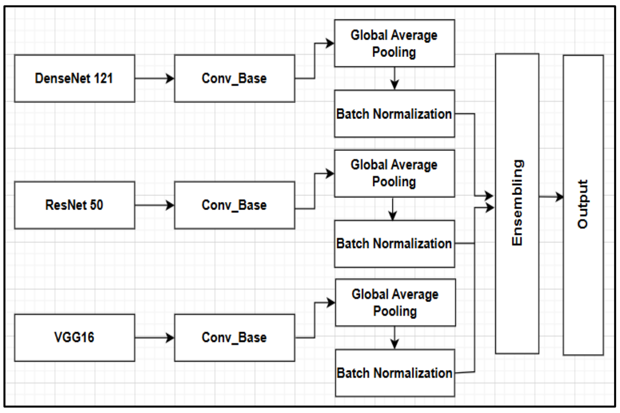

Fig-2: Methodology followed for Feature Ensemble Model.

First Features of whole 3 models were then concatenated into an ANN layer classifier, the one they had already, of which they were sliced away. Now three models’ conv_bases had a single ANN Classifier, which was to be trained. 30 Epochs at learning rate 2e-4 were trained to give a val_accuracy of 0.5856, further training improves it to 0.5965. Further training at 15 epochs at learning rate ’1e-3’ was done to prevent underfitting. Random Forests (‘val_accuracy’: 0.6106, ‘n_estimators’: 300, ‘max_features’: ‘log2’, ‘max_depth’: ‘None’, ‘min_samples_split’: 5, ‘min_samples_leaf’: 1) earns a good val_accuracy, due to n_estimators and depth being such a big number in these cases. ANN classifier top looks big, but RNN is working much deeper and with many models inside it, bagging upon which is performed to get the final results, which even further reduces any Noice or wrong data study. XGB (‘val_accuracy’: 0.6087, ‘n_estimators’: 200, ‘max_depth’: 7, ‘lr’: 0.1, ‘subsample’: 0.8, ‘colsample_bytree’: 0.6) performed pretty well, but, lower to Random Forests, because the data is not completely pure, and mistakes rectification technique actually started to be an obstacle. SVM (‘val_accuracy’: 0.6387, ‘C’: 1, ‘kernel’: ‘poly’, ‘gamma’: ‘scale’, ‘degree’: 3, ‘class_weight’: ‘balanced’) as usual, gave the best results amongst all as shown in Table 5. Due to its highly optimizable quality with a range of hyperparameters to tune with and capture the data in the best form. Note that, low C value and high degree, lead to such poor val_accuracy for SVM due to the very high flexibility created by this combination, that ruins the performance and the task.

Table-5: Validation Accuracy Comparison for Feature Ensemble Models

Model	Accuracy (%)
SVM	**63.87**
Random Forests	61.06
XG-Boost	60.87
ANN	58.56

** Highest accuracy is shown in bold*

IV. CONCLUSION AND FUTURE SCOPE

DenseNet-121 emerges as a highly effective model for fine-tuning on AffectNet for FER tasks, balancing accuracy and computational efficiency. SVM has proven to be an excellent classifier due to its extreme hyperparameter tuning options and one versus rest classifier, option of SVM is best which creates a unique SVM for each class. But along with that, it comes with its drawbacks and challenges of being very time-consuming and computationally expensive, to find the right hyper-parameters. Also, it cannot be fine-tuned or trained on the model just like ANN Classifiers can, by directly using backpropagation. Use of ‘Categorical_Hinge’ to copy those results have failed in case there are multiclass classification. XG-Boost and Ada-Boost like algorithms are very powerful, but if the data is wrong or noisy, then their very feature of focusing on mistakes can prove to lower their performances compared to light models like Random Forests, that just, builds individual models and collects their responses at the end, which reduces the noise influence by natural order. It can be concluded that choice of fine-tuning strategies is crucial for an effective use of pretrained model. In future researchers can explore Encoder-Decoder and Transformer Architectures along with various state of the methods to find the best ways to use and build them for pretrained model for the robust, efficient and scalable applications.

References

[1] Canal, F. Z., Müller, T. R., Matias, J. C., Scotton, G. G., de Sa Junior, A. R., Pozzebon, E., & Sobieranski, A. C. (2022). A survey on facial emotion recognition techniques: A state-of-the-art literature review. Information Sciences, 582, 593-617.

[2] Mahalle, V. S., Kandoi, N. M., & Patil, S. B. (2023, June). Transfer learning by fine-tuning pre-trained convolutional neural network architectures for image recognition. In International Conference on Data Science and Big Data Analysis (pp. 273-287). Singapore: Springer Nature Singapore.

[3] Yen, C. T., & Li, K. H. (2022). Discussions of different deep transfer learning models for emotion recognitions. IEEE Access, 10, 102860-102875.

[4] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.”

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[7] Öztürk, C., Taşyürek, M., & Türkdamar, M. U. (2023). Transfer learning and fine-tuned transfer learning methods' effectiveness analyse in the CNN-based deep learning models. *Concurrency and computation: practice and experience*, *35*(4), e7542.

[8] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

[9] Upadhyayula, V. P., & Amudha, V. (2024, May). Comparison of convolution neural network with support vector machine classifier for emotional face detection using machine learning. In *AIP Conference Proceedings* (Vol. 2853, No. 1, p. 020151). AIP Publishing LLC.

[10] Wiens, M., Verone-Boyle, A., Henscheid, N., Podichetty, J. T., & Burton, J. (2025). A tutorial and use case example of the eXtreme gradient boosting (XGBoost) artificial intelligence algorithm for drug development applications. *Clinical and Translational Science*, *18*(3), e70172.

[11] Lung, R. I., & Suciu, M. A. (2025). Log-Loss Optimization for Boosting a Nash Equilibrium Decision Tree. *Cybernetics and Systems*, 1-23.

[12] Kunapuli, G. (2023). *Ensemble methods for machine learning*. Simon and Schuster.

[13] Barreñada, L., Dhiman, P., Timmerman, D., Boulesteix, A. L., & Van Calster, B. (2024). Understanding overfitting in random forest for probability estimation: a visualization and simulation study. *Diagnostic and Prognostic Research*, *8*(1), 14.

[14] Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, *10*(1), 18-31.

[15] Rashad, M., Alebiary, D. M., Aldawsari, M., El-Sawy, A. A., & AbuEl-Atta, A. H. (2024). CCNN-SVM: automated model for emotion recognition based on custom convolutional neural networks with SVM. *Information*, *15*(7), 384.

[16] Wang, Y., Li, Y., Song, Y., & Rong, X. (2019). Facial expression recognition based on random forest and convolutional neural network. *Information*, *10*(12), 375.

[17] Mienye, I. D., & Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, *10*, 99129-99149.

[18] Thiab, A., Alawneh, L., & Mohammad, A. S. (2024). Contextual emotion detection using ensemble deep learning. *Computer Speech & Language*, *86*, 101604.