Journal of Theoretical and Applied Information Technology
15th April 2022. Vol.100. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FACIAL EMOTION RECOGNITION ON FER2013 USING VGGSPINALNET

BAMBANG EKO SANTOSO[1], GEDE PUTRA KUSUMA[2]

[1,2]Computer Science Department, BINUS Graduate Program – Master of Computer Science,

Bina Nusantara University, Jakarta, Indonesia, 11480

E-mail: [1]bambang.santoso001@binus.ac.id, [2]inegara@binus.edu

ABSTRACT

Facial Emotion Recognition is a part of human-computer interaction research whose results can be applied to society, ranging from social robots to accompany elderly to video games that adjust the level of difficulty based on the player's facial expressions. Many researchers attempt to create models for better accuracy with deep learning techniques, especially using Convolutional Neural Networks (CNN) that perform well in pattern recognition and image processing. This research adopts the state-of-the-art models in ImageNet and modifies the classification layer with SpinalNet and ProgressiveSpinalNet architecture to improve the accuracy. The classification is performed on the dataset FER2013, which is openly shared to the public in Kaggle contains over 35.000 face image datasets for seven distinct emotions. After finishing the training process and finetuning its hyperparameter, the resulting model with SpinalNet and ProgressiveSpinalNet outperformed all the existing single standalone model research on FER2013. One of the proposed architectures, VGGSpinalNet, has achieved the highest single network accuracy of 74.45%.

Keywords: *Facial Emotion Recognition, FER2013 Dataset, VGG Model, SpinalNet, ProgressiveSpinalNet*

1. INTRODUCTION

Recognition of facial expressions is one of the essential factors in recognizing human emotions [1]. This field is a part of human-computer interaction research which has a positive impact on society. This field research can have result to be implemented in the form of humanoid robots [2] that can interact with humans and respond based on human emotion for example helping in the child daycare. Furthermore, it can be used in healthcare to help detect the emotion of patients to indicate their mental health condition [3]. It can be used also for the entertainment industry such as video games that can adjust the flow of the game based on the player's facial expressions [4].

In addition to being neutral, basic human emotions are divided into six other types: anger, disgust, fear, happiness, sadness, and surprise [5]. The seven facial expressions become the classifications found in the existing datasets in general. There are several popular datasets for facial expression recognition research, such as the JAFFE [6], Cohn-Kanade Dataset (CK) [7], Extended Cohn-Kanade Dataset (CK+) [8], KDEF [9], AffectNet [10], Static Facial Expression in the Wild (SFEW) [11], FER2013 [12]. FER2013 was introduced at the International Conference on Machine Learning (ICML) and used frequently in research with topics of facial emotion recognition. Based on the literature review in previous facial expression recognition with the FER2013 dataset, we find that the highest accuracy was 76.82% [13] using ensembled CNN, while research using a single model without additional data reached 73.28% accuracy [14]. From this review, research on the FER2013 dataset the problem still to improve the accuracy performance of the model, especially on a single model. Many experiments use a variety of models in the experiment with FER2013, and the notable architecture is VGG [14]–[17], which reached state of the art a few times.

SpinalNet [18] uses the concept of a human somatosensory system to implement into the neural network architecture as a classification layer. Instead of an extensive network input on a fully connected layer, SpinalNet takes input gradually and considers local decisions like the spinal cord. SpinalNet and other biological architecture inspired ProgressiveSpinalNet [19]. ProgressiveSpinalNet has a gradient highway between the input to output layers, which solves the diminishing gradient problem. All the layers receive the input from the previous layer and the CNN layer output, and with this approach, all layers contribute to decision-making with the last layer.

Based on previous research done there is still a problem to be solved for further development, such as increasing accuracy in detecting emotion. In this research, we aim to improve the facial expressions accuracy that using FER2013 dataset with CNN models combined with SpinalNet architecture. We use the same VGG architecture and other notable models with modification on the classification layer replaced with SpinalNet and ProgressiveSpinalNet. As the result of this work, we achieved the highest single-model accuracy on FER2013 using the VGGSpinalNet model with a testing accuracy of 74.45%.

2. RELATED WORKS

The study of emotion has gained interest since the early 1970s, with Ekman and Friezen introducing six basic emotions: anger, disgust, fear, happiness, sadness, and surprise [5]. Later in 2004, Matsumoto and Ekman study about contempt as another universal primary emotion [20]. Ekman also created tools to support the study of emotion recognition explained in his book named Facial Action Coding System (FACS) [21]. Recent research on facial expression recognition using deep learning adopts commonly used methods such as VGG, DenseNet, Inception, ResNet, Caffe-ImageNet with FER2013 as the most widely used public dataset.

2.1 Facial Emotion Recognition (FER) Research Based on Single Standalone Model

Minae and Abdolrashidi conducted research [22] using the Attentional CNN method, which is a method that focuses on some facial regions, which are essential areas that are critical features in facial expression recognition. Due to the small number of classifications from the research, it is more effective to use CNN with a layer number of less than ten and using the attention method trained from the beginning can produce a satisfactory accuracy (70.02% in the FER2013 dataset and more than 90% in CK+, JAFFE, and FERG). They had tried to use more than 50 network layers. Still, the increase in accuracy was not significant, so it was concluded that attention on a neural network with a small number of layers could match the performance of deeper CNN.

GP Kusuma et al. researched facial emotion recognition [16] using a standalone-based CNN based on the Visual Geometry Group - 16 (VGG-16) classification model, which previously had a training process on ImageNet datasets after that finetuning process for facial expression classification. The modifications made to the VGG-16 model are replacing the VGG-16 classifier with a standard classifier which produces an accuracy of 69.40%. They have tried a combination of SGD, Adam, SWATS, but the best performance is obtained when using GAP as the final pooling layer.

Wang et al. conducted a study [3] to create facial expressions recognition implemented on a robot for medical patients. FER2013 was used as the primary dataset for training and to improve the accuracy. The trained model is retraining again with the NVIE dataset as additional data to FER2013. ResNet was used as a based model, which addresses the degradation problem of large neural networks by using a residual learning framework. The proposed architecture was to add the gate to each branch: the plain connection gate and the shortcut connection gate. The accuracy result on the FER2013 dataset was 71.8%.

The research conducted by Kathryn et al. [23] aims to recognize identity and facial expressions using the DenseNet model with one convolution layer, three dense blocks, and 1 FC layer. Each dense block contains three densely connected convolution layers with kernel size 3, stride 1, and padding 1. The research resulted in an accuracy of 63.5% for the FER2013 dataset and 85.4% for the KDEF dataset. Based on this FER2013 which was taken in a wild setting, versus KDEF, which was taken in the experimental setup, the dataset taken in a wild setting is more challenging to gain higher accuracy from training.

Khaireddin and Chen, with their research [14], used a standalone model based on VGG, resulting in a state-of-the-art performance for a single model for the FER2013 dataset. Their model is based on modified VGG11 with batch normalization. The model was evaluated using validation accuracy and tested using standard ten-crop averaging. To increase the accuracy, the model retrained again with both train and validation datasets as training data resulting accuracy of 73.28%.

Based on the study of a single standalone model for FER2013, we can see that the best testing accuracy for single model experiments was achieved using VGGNet. The classifier for that model only used one single FC layer.

2.2 Facial Emotion Recognition (FER) Research Based on Ensemble Model

Chao Li et al. research [24] used the FER2013 and JAFFE datasets based on Tang's network structure in the FER2013 competition at Kaggle and Caffe ImageNet. In his research, Li conducted separate training to take initialization parameters in Multi-Network Fusion (MNF). After that, finetuning of the MNF parameters was carried out for the

adjustment process and using L2-SVM for classification to replace Softmax Classification. The inadequate images from FER2013 and JAFFE make it easy to overfit when using a complex CNN model. The researcher performed data augmentation to multiply the dataset by randomly duplicating the image, removing 3 pixels from various sides, rotating 45 degrees, enlarging 1.2 times, and cropping the image into 42x42 size. The original image also needs to be dimmed so that the accuracy is further increased by the Histogram equalization (Hist-eq) process. Hist-eq is a good technique for normalizing grayscale images, making it easier to distinguish between a background image and a face. The first CNN structure based on Tang consists of 3 CNN networks and a max-pooling layer, and all layers are connected and then using L2-SVM to classify facial expressions into seven types. The second CNN based on Caffe-ImageNet sequentially consists of 2 CNN network layers, max-pooling layer and LRN layer, 2 CNN layers, max-pooling layer, a CNN layer, all connected by one layer. The combined two CNN structures with the L2-SVM classifier produced an accuracy of 70.3%.

In their research [15], Martin tried to compare the performance of shallow CNN with few layers with modern CNN (VGG, Inception, ResNet) on the FER2013 dataset. They confirmed that modern CNN performed better than shallow CNN. The results of his research stated that conducting a CNN ensemble of 8 models produced an accuracy of 75.2% without using additional datasets and face registration. The problem encountered in the study was caused by the dataset used. The amount of data was still less than the standard amount of data used in deep learning in general.

Nguyen et al., doing research [17], used Multi-Level Convolutional Neural Networks (MLCNN), whose model consisted of 18 layers CNN based on VGG net. Nguyen used high-level and mid-level features taken from blocks 2-4 of the network in his model. The first block is not used because the first block focuses on unimportant things like the background and hair. The ensemble model used consists of 3 MLCNN and three layers that are all connected. The first two layers have 512 hidden units, and the last is a 7-way SoftMax classifier resulting in an accuracy of 74.09%.

Based on the study of the ensemble model for FER2013, we can see that ensemble models have higher testing accuracy but use more resources for computing and more time for processing. Also, the important thing is the performance of the ensemble model is greatly affected by the single model. In conclusion, improving single model accuracy will improve both methods.

3. THEORY AND METHODS

3.1 VGGNet

VGG [25] is a convolution neural network architecture proposed by Karen Simonyan and Andrew Zisserman of the University of Oxford in 2014. This architecture is the 1'st runner-up of ImageNet Large Scale Visual Recognition Challenge 2014(ILSVR2014) in the classification task, while the winner is GoogleNet [26]. VGG addresses the critical aspect of CNN, which is depth. Common VGG model name is based on weight layer, VGG16 has 16 weight layers (13 convolution layers and three fully connected layers), and VGG19 has 19 weight layers (16 convolution layers and three fully connected layers).

VGGNet, which achieves the highest accuracy as a single model on FER2013 public test data [14] shown in Figure 1, consists of 4 Convolution Blocks as Feature Layers and 2 FC Layers with DropOut and ReLU layer and end with 1 FC Layer.

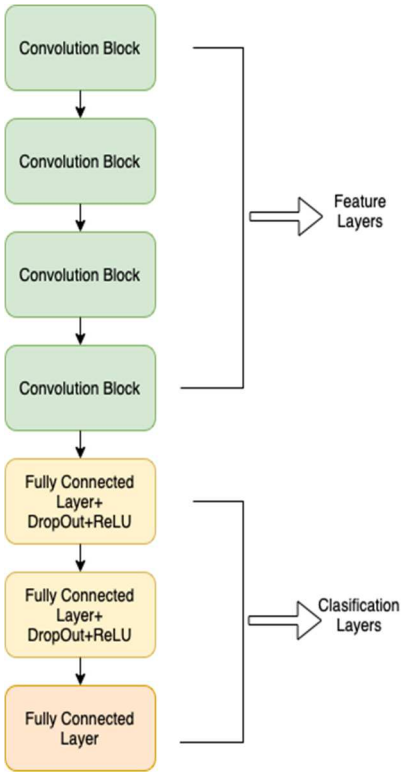

Figure 1: VGGNet Architecture

The Convolution Block consists of 2 convolution layers with ReLU and batch normalization and ends with max pooling, shown in Figure 2. Batch normalization is added on VGG to speed up the

training and avoid internal covariate shift.

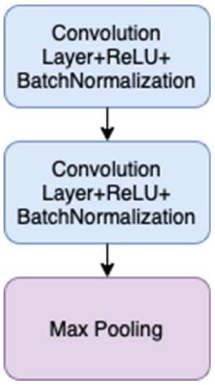

Figure 2: Convolution Block Architecture

3.2 EfficientNetV2

EfficientNetV2, created by Tan et al. [27], is a new architecture based on EfficientNet [28] architecture with faster training speed and more efficiency. EfficientNetV2 proposes Fused-MBConv that replaces depthwise 3x3 convolution and 1x1 convolution in MBConv with a regular 3x3 convolution. However, when all MBConv replaced with Fused-MBConv will slow the training speed, the authors use neural architecture search to search for the best combination. MBConv or Inverted Residual Block is a residual block used for image models that use an inverted structure for efficiency purposes. It was first introduced on MobileNetV2 CNN architecture [29]. The complete architecture of MBConv and Fused-MBConv is shown in Table 1. SE Layer or Squeeze and Excitation Block [30] is a layer that adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels. Depthwise Convolution [31] is a type of convolution where a single convolution filter is applied for each input channel.

Table 1: MBConv and Fused-MBConv architecture

No.	MBConv	Fused-MBConv
1	Conv 1x1	Conv 1x1
2	SE Layer	SE Layer
3	Depthwise Conv 3x3	Conv 3x3
4	Conv 1x1	

EfficientNetV2-S is a variant of EfficientNetV2. Its architecture started with convolution layer 3x3 followed by three blocks of Fused-MBConv and 3 MBConv blocks. It ended with convolution 1x1 layer, pooling layer, and FC layer. The efficientNetV2-S complete architecture is described in Table 2.

Table 2: EfficientNetV2-S Architecture

Stage	Operator	Stride	Channels	Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv1, k3x3	2	48	4
3	Fused-MBConv1, k3x3	2	64	4
4	MBConvV4, k3x3 SE0.25	2	128	6
5	MBConvV6, k3x3 SE0.25	1	160	9
6	MBConvV6, k3x3 SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

3.3 Vision Transformers

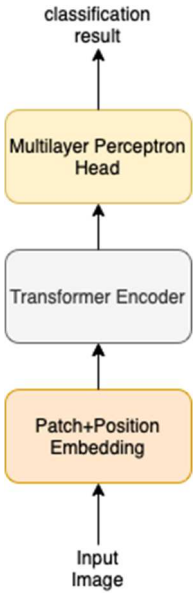

Figure 3: Vision Transformer Architecture

Vision Transformer is a new model for image classification that use architecture like Transformer architecture designed for natural language processing task [32]. The main component of the architecture is shown in Figure 3, where the structure consists of Patch + Position Embedding, Transformer Encoder, and Multilayer Perceptrons Head. Patch + Position Embedding block architecture handles 2D images input by reshaping the image into a sequence of flattened 2D parches like Figure 4. The image is split into fixed-size patches, and the image dimensions can be divided by the patch size. The patches are put into a single vector added position embedding to retain the positional information. The joint embedding becomes an input to the transformer encoder.

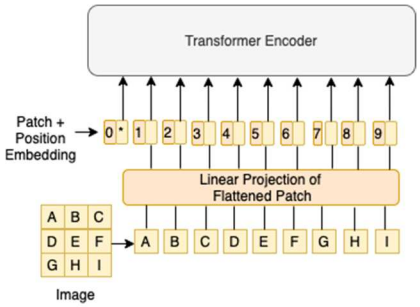

Figure 4: Patch + Position Embedding

The Transformer Encoder consists of Multi-Head Self Attention Layer, and Multi-Layer Perceptron Block is shown in Figure 5. Layer Normalization is added before every block, and there is a residual connection after every block. Multiplayer Perceptron Head as the last layer in vision transformer consists of one hidden layer at the pre-training time and a single linear layer at finetuning time.

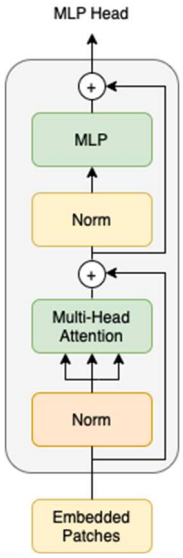

Figure 5: Transformer Encoder

ViT-L/32(21k) is a vision transformer variant with large size of 24 layers with the 32x32 input patch size and pre-trained weight from ImageNet 21k classes. If the patch size is smaller, the computational source becomes more expensive.

3.4 SpinalNet

Kabir, H. M., et al., inspired by the human somatosensory system and the spinal cord, created SpinalNet [18]. Typical neural network architecture will consist of a large number of parameters, resulting in high computational demand. This concept is similar to the input to the brain and how neural input to the brain work. SpinalNet takes inputs gradually. In the proposed SpinalNet, the structure of hidden layers allocates to three sectors: input row, intermediate row, and output row. The intermediate row of the SpinalNet contains a few neurons. The role of input segmentation is in enabling each hidden layer to receive a part of the inputs and outputs of the previous layer. In SpinalNet, the number of incoming weights in a hidden layer is significantly lower than in a traditional neural network. As all layers of the SpinalNet directly contribute to the output row, the vanishing gradient problem does not exist. The research used VGG, DenseNet, and ResNet as a based model and combined it with SpinalNet as a classification layer, successfully increasing the accuracy of each model for QMNIST, Kuzushiji-MNIST, EMNIST (Letters, Digits, and Balanced), STL-10, Bird225, Fruits 360, and Caltech101 datasets.

3.5 ProgressiveSpinalNet

Praveen Chopra was inspired by SpinalNet and other biological architecture, resulting in architecture called ProgressiveSpinalNet [19]. The size of each layer progressively increased in each layer of the FC. This gradual increase did not make the network very bulky as the increment is equal to the size of the first layer output. This many numbers neurons was added into each step of the FC network. In the proposed ProgressiveSpinalNet architecture, the output size is kept the same for each FC layer. This concept made the network design simple and more effectively manageable. The performance of the ProgressiveSpinalNet was tested on various models along with the transfer learning models. The performance ProgressiveSpinalNet on both models is better than the SpinalNet or at par with SpinalNet.

4. PROPOSED DEEP LEARNING ARCHITECTURE

This research proposes new architecture by combining architecture that already achieves the best performance on ImageNet like EfficientNetV2, ViT-L/32(21k), and VGGNet that reach the highest accuracy single model on FER2013 with architecture that improve the classification process like SpinalNet and ProgressiveSpinalNet.

4.1 VGGSpinalNet

VGGSpinalNet architecture, as shown in Figure 6, is a combination of VGGNet and SpinalNet. The Classification Layers on VGGNet is replaced with SpinalNet architecture. Feature Layers consists of 4 convolution blocks with 64, 128, 256, and 512 tensor width sizes, respectively. The output of the last

convolution blocks will be changed to 1 dimension tensor of 2048 and split into two, with the 1'st half going to Spinal Layer 1 and 3 and the 2'nd half going to Spinal Layer 2 and 4 with the size of 1024 each. The output of the Spinal Layer goes to the next Spinal Layer and concatenates with the other Spinal Layers result. Spinal Layer formed by dropout layer with parameter 0.5, linear layer with tensor width 1024 input and 512 output, except for first Spinal Layer has 512 tensor width input, batch normalization layer, and ReLU layer. The output, FC Out, consists of the dropout and linear layers having 2048 tensor width input and output seven as the number of classes.

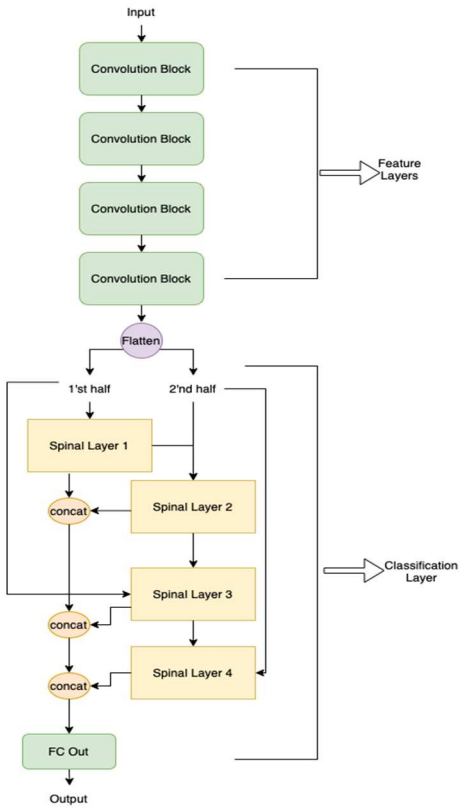

Figure 6: VGGSpinalNet Architecture

4.2 VGGProgressiveSpinalNet

VGGProgressiveSpinalNet architecture, as shown in Figure 7, is a combination of VGGNet and ProgressiveSpinalNet. The last FC Out Layers on VGGNet is replaced with ProgressiveSpinalNet architecture. Feature Layers consists of 4 convolution blocks with 64, 128, 256, and 512 tensor width sizes, respectively. The output of the last convolution blocks will be changed to 1 dimension tensor of 2048. The output of the last convolution blocks will go into Dropout Layer and Spinal Layer. The Spinal Layer consists of a linear layer and a ReLU layer. The output of the Spinal Layer has 512 tensor widths and is concatenated with Dropout Layer, which has the size 2048. The concatenated result will go into the next Spinal Layer and make the following Spinal Layer tensor width input grew by 512 each repetition and the output always 512 widths size and will concatenate with previous concatenation result. There are 4 Spinal Layers, and the last concatenated node will go to FC out, a linear layer with the input of concatenated tensor having tensor width size 4096 and output seven as the number of classes.

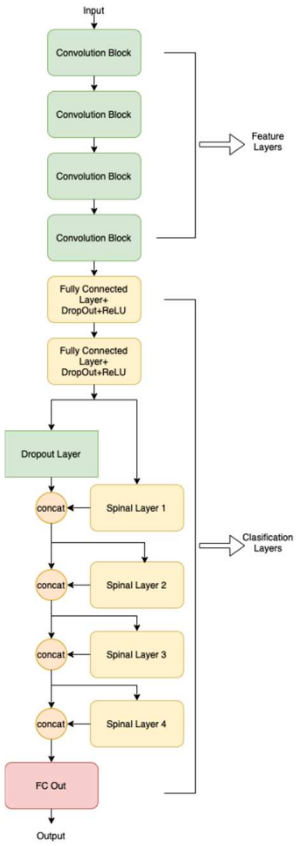

Figure 7: VGGProgressiveSpinalNet Architecture

4.3 EfficientNetV2-S-SpinalNet

EfficientNetV2-S-SpinalNet architecture, as shown in Figure 8, is a combination of EfficientNetV2-S and SpinalNet. EfficientNetV2-S takes three-channel colors as input and needs to be changed to 1 channel because FER2013 consists of a grayscale image, which is one channel. EfficientNetV2-S consists of Convolution Layer, 3 Fused MBConv Layer, 3 MBConv Layer, 1 Convolution and Pooling Layer, with output FC Layer. In this proposed architecture, the last FC Layer is replaced with SpinalNet architecture. The

Journal of Theoretical and Applied Information Technology
15th April 2022. Vol.100. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

output of EfficientNetV2-S is 1792 tensor width size split into two, with the 1'st half going to Spinal Layer 1, and 3 and 2'nd half going to Spinal Layer 2 and 4 with the size of 896 each. The output of the Spinal Layer goes to the next Spinal Layer and concatenates with the other Spinal Layers result. Spinal Layer formed by dropout layer with parameter 0.5, linear layer with tensor width 1408 input and 512 output, except for first Spinal Layer has 896 tensor input, batch normalization layer, and ReLU layer. FC Out as the output layer consists of dropout and linear layers with size 2048 and output seven as the number of classes.

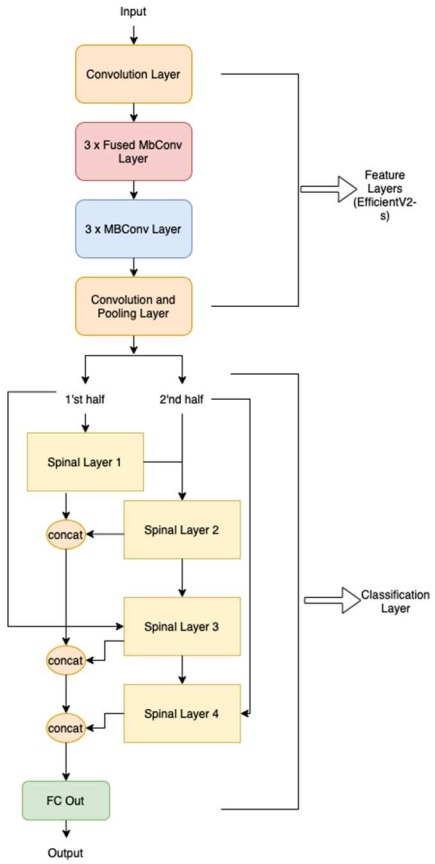

Figure 8: EfficientNetV2-S-SpinalNet Architecture

4.4 ViT-L/32(21k)-SpinalNet

ViT-L/32(21k)-SpinalNet Architecture, as shown in Figure 9, is a combination of ViT-L/32(21k) with SpinalNet. ViT-L/32(21k) consists of Patch and Embedding Position Layer, Transformer Encoder Layer, and Multilayer Perceptron Head. In this proposed architecture, the classifier layer, a FC Layer, is replaced with SpinalNet architecture. The output of ViT-L/32(21k) is 1024 tensor width size and split into two, with the 1'st going to Spinal Layer 1 and 3 and 2'nd going to Spinal Layer 2 and Spinal Layer 4 with the size of 512 each. The output of the Spinal Layer goes to the next Spinal Layer and concatenates with the other Spinal Layers result. Spinal Layer formed by dropout layer with parameter 0.5, linear layer with tensor width size 1024 for input and 512 output, except for first Spinal Layer has 512 tensor width input, batch normalization layer, and ReLU layer. The output, FC Out, consists of the dropout and linear layers with tensor width 2048 and output seven as the number of the classes.

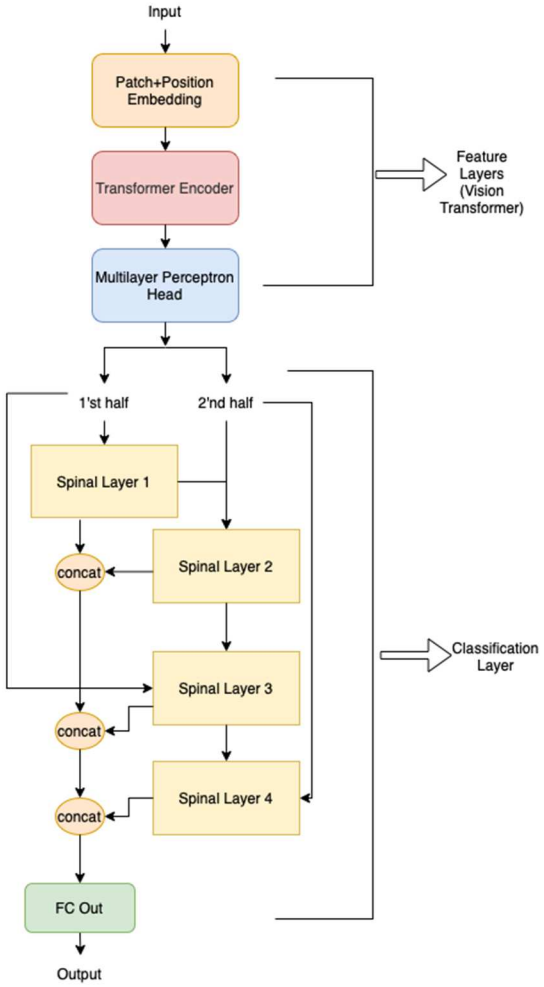

Figure 9: ViT-L/32(21k)-SpinalNet Architecture

5. EVALUATION

5.1 Dataset

The FER2013 dataset is an open-source dataset created by Pierre-Luc Carrier and Aaron Courville for competition on the platform Kaggle [33]. FER2013 contains 35887 grayscale 48x48 pixel images with a label containing a human face with seven different emotions: anger, neutral, disgust,

fear, happiness, sadness, and surprise. Figure 10 shows the samples of the FER2013 dataset. The training set consists of 28,709 pictures. The public test set used for the leaderboard consisting of 3,589 pictures will be used as validation data. The final test set, used to determine the competition's winner, consists of another 3,589 images used as test data.

Figure 10: FER2013 sample images

5.2 Preprocessing

5.2.1 VGGSpinalNet and VGGProgressiveSpinalNet Preprocessing

Preprocessing for VGGSpinalNet and VGGProgressiveSpinalNet has the same process because both have the same based model. Each model has a significant amount of data augmentation in training. Data augmentation for the training process includes rescaling the images up to ± 20 % of their original scale, horizontally and vertically shifting the image by up to 20% more or 20% less of its size and rotating it up to ± 10 degrees. Each of the techniques is applied randomly with a probability of 50%. The images are then ten cropped to size 40×40, and random portions of each crop are erased with a chance of 50 %. Each crop is normalized by dividing each pixel by 255. Preprocessing for validation and testing are the input images ten cropped to size 40x40, and each crop is normalized by dividing each pixel by 255.

5.2.2 EfficientNetV2-S-SpinalNet Preprocessing

EfficientNetV2-S-SpinalNet preprocessing for the training process consists of a random image crop to size 40x40 and a random horizontal flip. Preprocessing for validation and testing process are the input images ten cropped to size 40x40.

5.2.3 ViT-L/32(21k)-SpinalNet Preprocessing

Preprocessing for ViT-L/32(21k)-SpinalNet model to training, validation, and testing process all using the same process. Those processes consist of resizing the image using ESRGAN [34] to 380x380 pixel and rescaling it to 224x224 pixels using the bicubic interpolation method.

5.3 Experimental Design

For each proposed architecture, there will be a training, validation, and testing process. The training process uses the Pytorch framework on Google Colab Pro Services, specifications described in Table 3.

Table 3: Google Colab Pro Specification

No	Specification	Value
1	GPU Type	Nvidia Tesla P100 or T4
2	GPU Memory	16 GB
3	CPU	2 x vCPU
4	RAM	24 GB

5.3.1 VGGSpinalNet and VGGProgressiveSpinalNet Training and Validation

The training process for VGGSpinalNet and VGGProgressiveSpinalNet starts by training the basic VGGNet from scratch without a pre-training weight model with the hyperparameter setup in Table 4. VGGNet weight results will be transferred to VGGSpinalNet and VGGProgressiveSpinalNet for further training.

Table 4: VGGNet Hyperparameter Setup

No	Hyperparameter	Value
1	Learning Rate	0.01
2	Optimizer	SGD with Nesterov, Ranger21, Ranger, Adam
3	Learning Rate Scheduler	ReduceLRonPlateau, CosineAnnealingLR, Step Decay
4	Number of Epoch	200
5	Batch Size	64

To train VGGSpinalNet and VGGProgressiveSpinalNet, start loading the weight for all layers with the same name from VGGNet to both models. All the layer that has been weight transferred optionally need to be frozen by setting the required grad to false for weight and bias parameter. For training both models, this experiment will use a variety of optimizers, learning rate schedulers, and batch size is shown in Table 5.

Table 5: VGGSpinalNet and VGGProgressiveSpinalNet Hyperparameter Setup

No	Hyperparameter	Value
1	Learning Rate	0.001
2	Optimizer	SGD with Nesterov, Ranger21, Ranger, Adam
3	Learning Rate Scheduler	ReduceLRonPlateau, CosineAnnealingLR, Step Decay
4	Number of Epoch	200
5	Batch Size	64

5.3.2 EfficientNetV2-S-SpinalNet Training and Validation

The training process for EfficientNetV2-S-SpinalNet starts by training the basic EfficientNetV2-S without pre-trained model weight with the hyperparameter setup in Table 6. Ranger Optimizer [35], one of the optimizers used in this experiment, is a combination of RAdam, LookAhead [36], and Gradient Centralization. The model also replaces the Swish activation function, a default activation function for EfficientNetV2, with the Mish activation function. Mish is a self-regularized non-monotonic activation function [37] that has better performance than ReLU and Swish.

Table 6: EfficientNetV2-S Hyperparameter Setup

No	Hyperparameter	Value
1	Learning Rate	0.005
2	Optimizer	SGD with Nesterov, Ranger21, Ranger, Adam
3	Learning Rate Scheduler	ReduceLRonPlateau, CosineAnnealingLR, Step Decay
4	Number of Epoch	2000
5	Batch Size	128
6	Activation Function	Swish, Mish

All layers from the EfficientNetV2-S model, which has been finished training, will be frozen by setting the required grad to false. The classifier layer, which is a linear layer, will be replaced with SpinalNet Architecture. The training use hyperparameter showed in Table 7.

Table 7: EfficientNetV2-S-SpinalNet Hyperparameter Setup

No	Hyperparameter	Value
1	Learning Rate	0.005
2	Optimizer	SGD with Nesterov, Ranger21, Ranger, Adam
3	Learning Rate Scheduler	ReduceLRonPlateau, CosineAnnealingLR, Step Decay
4	Number of Epoch	1000
5	Batch Size	128
6	Activation Function	Swish, Mish

5.3.3 ViT-L/32(21k)-SpinalNet Training and Validation

The training process for the Vision Transformer Model with the limited resource of Google Colab is quite tricky because GPU RAM is not enough, and the batch becomes too small and makes the training unsuccessful. To make the training feasible for ViT-L/32(21k) will be using Hugging Face Library [38]. ViT-L/32(21k)-SpinalNet start by training basic ViT-L/32(21k) model using pre-trained model 'google/vit-large-patch32-224-in21k' with the hyperparameter setup in Table 8.

Table 8: ViT-L/32(21k) Hyperparameter Setup

No	Hyperparameter	Value
1	Learning Rate	0.00002
2	Optimizer	Ranger
3	LR Decay	Decay Every=5, Decay Rate=0.9
4	Number of Epoch	100
5	Batch Size	Training=19, Evaluation and Testing=1

The weight from ViT-L/32(21k) training result will be transferred to the ViT-L/32(21k)-SpinalNet and freeze the weight after that. The training will use hyperparameters setup in Table 9.

Table 9: ViT-L/32(21k)-SpinalNet Hyperparameter Setup

No	Hyperparameter	Value
1	Learning Rate	0.0001
2	Optimizer	Ranger21
3	Ranger21's Weight Decay	0.0005
4	Ranger21's Warm Up	True
5	Ranger21's Look Ahead Active	True
6	Ranger21's Norm Loss	Active=True, Factor=6e-4
7	Ranger21's Warm Down	Active=True, Start_pct=0.50, min LR=3e-6
8	Ranger21's Momentum	Type='pnm', factor=1.0, momentum=0.9
9	Number of Epoch	50
10	Batch Size	Training=19, Evaluation and Testing=1

5.3.4 Testing and Measure Performance

All models will be measured for accuracy using the FER2013 private data. In addition to using the overall accuracy calculation to evaluate the performance of the models that have been produced, a multiclass confusion matrix is used. To measure the model's performance, precision, recall, and F-Score calculations with the macro average [39]. The row elements in the confusion matrix are the number of datasets for each label. In contrast, the column elements in the confusion matrix are the predicted results of a model on the dataset in the form of numeric values.

Precision called positive predictive value is a comparison calculation of the number of true positives compared to all positive data. Recall, also known as sensitivity, is a comparison calculation of the number of true positives compared to the sum of all true data. F-Score is the harmonic mean of precision and recall. The macro average will treat all classes equally make the class with less data is also essential.

5.4 Experimental Result and Discussion

5.4.1 VGGSpinalNet Experiment Result

For the training and validation process, the optimum hyperparameter setup is shown in Table 10 using Ranger21 optimizer [40], which combination of AdamW with the look ahead and other improvements for training methods like Learning Scheduler and Gradient Clipping.

Table 10: Optimum Hyperparameter Setup for Training VGGSpinalNet

No	Hyperparameter	Value
1	Learning Rate	0.001
2	Optimizer	Ranger21
3	Ranger21's Weight Decay	0.0005
4	Ranger21's Warm Up	True
5	Ranger21's Look Ahead Active	True
6	Ranger21's Norm Loss	Active=True, Factor=6e-4
7	Ranger21's Warm Down	Active=True, Start_pct=0.50, min LR=3e-6
8	Ranger21's Momentum	Type='pnm', factor=1.0, momentum=0.9
9	Number of Epoch	200
10	Batch Size	64
11	Freeze layer from pretrain model	True

The accuracy and loss statistic for VGGSpinalNet is shown in Figure 11. The accuracy and loss reflected on that figure only until 69 epochs iteration because there is no improvement in the validation accuracy.

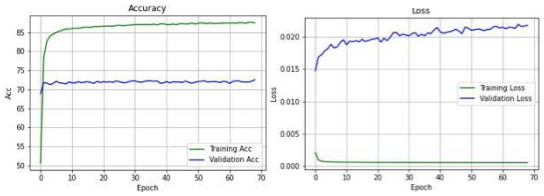

Figure 11: Training and Validation Accuracy and Loss for VGGSpinalNet

Figure 12 shows the confusion matrix result for the VGGSpinalNet training result with testing overall accuracy of 74.45%, macro average precision result 0.740, macro average recall result 0.727, and macro average F-score 0.733.

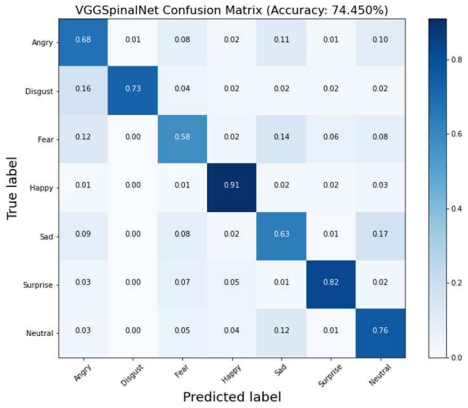

Figure 12: VGGSpinalNet Confusion Matrix Result

5.4.2 VGGProgressiveSpinalNet Experiment Result

For model VGGProgressiveSpinalNet, there is a finetuning process to increase the accuracy after the training process is finished. Finetuning process is done by using a lower learning rate to get optimum accuracy. The optimum hyperparameter configuration for the training is shown in Table 11.

Table 11: Optimum Hyperparameter Setup for Training VGGProgressiveSpinalNet

No	Hyperparameter	Value
1	Learning Rate	0.0001
2	Finetuning's Learning Rate	0.00001
3	Optimizer	SGD with Nesterov
4	SGD's Weight Decay	0.0001
5	SGD's Momentum	0.9
6	LR Scheduler	CossineAnnealingLR
7	Number of Epoch for Training	200
8	Number of Epoch for Finetuning	20
9	Batch Size	64
10	Freeze layer from pretrain model	False

The accuracy and loss statistic for training model VGGProgressiveSpinalNet is shown in Figure 13. The accuracy and loss reflected on that figure only until 109 epochs iteration because there is no improvement in the validation accuracy.

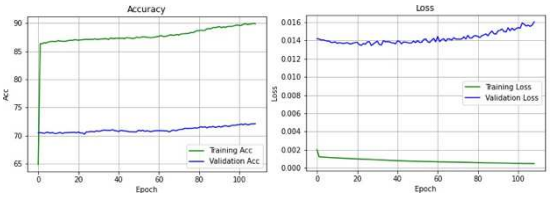

Figure 13: Training and Validation Accuracy and Loss for VGGProgressiveSpinalNet

The accuracy and loss statistic for training and validation to finetune the VGGProgressiveSpinalNet model is shown in Figure 14. The accuracy and loss reflected on that figure only until the 14 epochs iteration because there is no improvement in the validation accuracy after that.

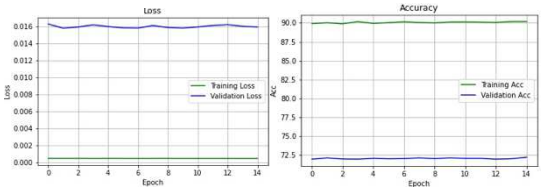

Figure 14: Training and Validation Accuracy and Lost for Finetuning VGGProgressiveSpinalNet

Figure 15 shows the confusion matrix result for the VGGProgressiveSpinalNet training with testing overall accuracy of 74.39%, macro average precision result of 0.740, macro average recall result of 0.725, and macro average F-score of 0.732.

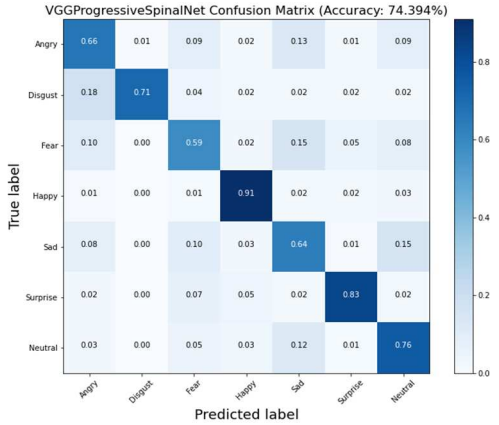

Figure 15: VGGProgressiveSpinalNet Confusion Matrix Result

5.4.3 EfficientNetV2-S-SpinalNet Experiment Result

EfficientNetV2-S-SpinalNet result is based on the optimum hyperparameter setup that shown in Table 12 with the Ranger21 optimizer.

Table 12: Optimum Hyperparameter Setup for Training EfficientNetV2-S-SpinalNet

No	Hyperparameter	Value
1	Learning Rate	0.005
2	Optimizer	Ranger21
3	Ranger21's Weight Decay	0.0005
4	Ranger21's Warm Up	True
5	Ranger21's Look Ahead Active	True
6	Ranger21's Norm Loss	Active=True, Factor=6e-4
7	Ranger21's Warm Down	Active=True, Start_pct=0.72, min LR=3e-5
8	Ranger21's Momentum	Type='pnm', factor=1.0, momentum=0.9
9	Number of Epoch	1000
10	Batch Size	128
11	Freeze layer from pretrain model	True

The accuracy and loss statistic for the training model EfficientNetV2-S-SpinalNet is shown in Figure 16. The accuracy and loss reflected on that figure only until the 18 epochs iteration because there is no improvement in the validation accuracy after that.

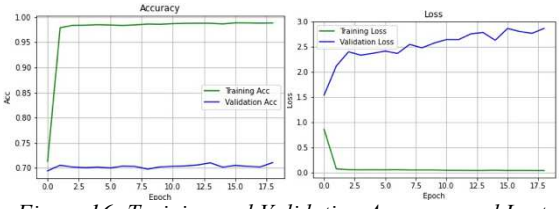

Figure 16: Training and Validation Accuracy and Lost for EfficientNetV2-S-SpinalNet

Figure 17 shows the confusion matrix result for the EfficientNetV2-S-SpinalNet training with testing overall accuracy of 72.11%, macro average precision result of 0.720, macro average recall result of 0.704, and macro average F-score of 0.712.

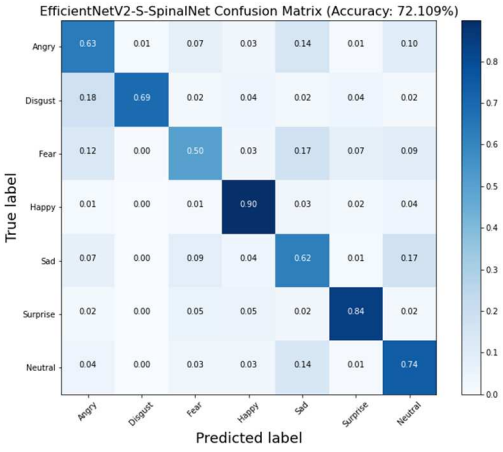

Figure 17: EfficientNetV2-S-SpinalNet Confusion Matrix Result

5.4.4 ViT-L/32(21k)-SpinalNet Experiment Result

ViT-L/32(21k)-SpinalNet result is based on the optimum hyperparameter setup that shown in Table 13.

Table 13: Optimum Hyperparameter Setup for Training ViT-L/32(21k)-SpinalNet

No	Hyperparameter	Value
1	Learning Rate	0.0001
2	Optimizer	Ranger21
3	Ranger21's Weight Decay	0.0005
4	Ranger21's Warm Up	True
5	Ranger21's Look Ahead Active	True
6	Ranger21's Norm Loss	Active=True, Factor=6e-4
7	Ranger21's Warm Down	Active=True, Start_pct=0.50, min LR=3e-6
8	Ranger21's Momentum	Type='pnm', factor=1.0, momentum=0.9
9	Number of Epoch	50
10	Batch Size	Training=19, Validation=1
11	Freeze layer from pretrain model	True

The accuracy and loss statistic for training model ViT-L/32(21k)-SpinalNet is shown in Figure 18. The accuracy and loss reflected on that figure only until the nine epochs iteration because there is no improvement in the validation accuracy after that.

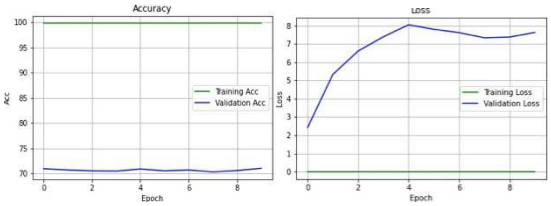

Figure 18: Training and Validation Accuracy and Lost for ViT-L/32(21k)-SpinalNet

Figure 19 shows the confusion matrix result for ViT-l/32(21k)-SpinalNet training result with testing overall accuracy 71.60%, macro average precision result 0.718, macro average recall result 0.702, and macro average F-score 0.710.

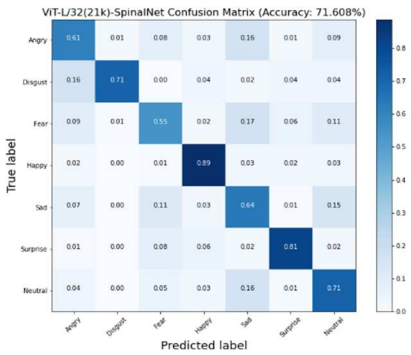

Figure 19: ViT-L/32(21k)-SpinalNet Confusion Matrix Result

5.4.5 Performance Comparison on FER2013 for Single Standalone Model

The resulting model from this experiment achieves the best single standalone model for FER2013 private test data with an accuracy of 74.45%. Based on the result and previously reported experiment [14]–[17], the VGG model variant performs better for FER2013 private test data than the other models. The latest model architecture like EfficientNetV2 and Vision Transformer variant model performs well on ImageNet dataset having overfitted on FER2013 that shown at training accuracy on FER2013 reach 98-99%, but the validation accuracy on FER2013 only achieves 70-71%. This overfitting problem is because both models are complex models that do not perform well on FER2013 which is a lack of data mentioned by previous research [24]. The small model works well on FER2013, shown by efficientNetV2-S and VGGNet [14], like VGG11.

Table 14 summarizes the performance of reported models on the FER2013 dataset and models trained on this experiment. All models trained on this work perform better than estimated human performance (around 65.5%). The resulting model from this work achieves state-of-art performance with accuracy 74.45% better than the previous best reported single-based model [14], which achieves 73.28% accuracy.

Table 14: FER2013 Testing Accuracy Comparison

Related Works	Method	Testing Accuracy
[23]	DenseNet	63.5%
[16]	VGG-16	69.40%
[22]	Attention CNN	70.02%
[24]	MNF CNN+L2 SVM	70.3%
Our Model	ViT-L/32(21k)	71.27%
Our Model	ViT-L/32(21k)-SpinalNet	71.60%
[3]	ResNet with Gate Implementation	71.8%
Our Model	EfficientNetV2-S	71.83%
Our Model	EfficientNetV2-S-SpinalNet	72.11%
[14]	VGGNet	73.28%
Our Model	VGGProgressiveSpinalNet	74.39%
Our Model	VGGSpinalNet	74.45%

6. CONCLUSION AND FUTURE WORK

In this work, we have contributed to solving the problem to increase the accuracy in facial expression recognition on FER2013 with our model, VGGSpinalNet achieved the best testing accuracy (74.45%). The best result was achieved with transfer learning from VGGNet to VGGSpinalNet with freezing the transferred weight during training.

Ranger21 optimizer having learning rate 0.001 on training process for 200 epochs.

Future work that can be done is to the architecture model VGGSpinalNet by modifying the architecture furthermore to improve the accuracy. Other methods like SVM and Multilevel CNN [17] can be added to the VGGSpinalNet. Another work that can be done is to train using the Vision Transformer model with capable hardware resources to gain better results. Ensemble models consisting of VGGSpinalNet and other models can also be used in subsequent future work to improve the accuracy furthermore.

REFERENCES:

[1] B. Fasel and J. Luettin, "Automatic facial expression analysis: A survey," *Pattern Recognition*, vol. 36, no. 1. Elsevier Ltd, pp. 259–275, Jan. 01, 2003. doi: 10.1016/S0031-3203(02)00052-3.

[2] O. Tutsoy, F. Göngör, D. Barkana, and H. Kose, "AN EMOTION ANALYSIS ALGORITHM AND IMPLEMENTATION TO NAO HUMANOID ROBOT," *The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)*, vol. 1, pp. 316–330, Jan. 2017.

[3] F. Wang, H. Chen, L. Kong, and W. Sheng, "Real-time Facial Expression Recognition on Robot for Healthcare," Apr. 2018, pp. 402–406. doi: 10.1109/IISR.2018.8535710.

[4] J. v. Moniaga, A. Chowanda, A. Prima, Oscar, and M. D. Tri Rizqi, "Facial Expression Recognition as Dynamic Game Balancing System," *Procedia Computer Science*, vol. 135, pp. 361–368, Jan. 2018, doi: 10.1016/J.PROCS.2018.08.185.

[5] P. Ekman and W. v. Friesen, "Constants across cultures in the face and emotion," *Journal of Personality and Social Psychology*, vol. 17, no. 2, pp. 124–129, Feb. 1971, doi: 10.1037/H0030377.

[6] F. Y. SHIH, C.-F. CHUANG, and P. S. P. WANG, "PERFORMANCE COMPARISONS OF FACIAL EXPRESSION RECOGNITION IN JAFFE DATABASE," *http://dx.doi.org/10.1142/S0218001408006284*, vol. 22, no. 3, pp. 445–459, Nov. 2011, doi: 10.1142/S0218001408006284.

[7] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," *Proceedings - 4th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2000*, pp. 46–53, 2000, doi: 10.1109/AFGR.2000.840611.

[8] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, pp. 94–101, 2010, doi: 10.1109/CVPRW.2010.5543262.

[9] D. Lundqvist, A. Flykt, and A. Ohman, "The Karolinska directed emotional faces (KDEF)," *CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet*, pp. 91–630, 1998.

[10] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild," *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 18–31, Jan. 2019, doi: 10.1109/TAFFC.2017.2740923.

[11] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2106–2112, 2011, doi: 10.1109/ICCVW.2011.6130508.

[12] I. J. Goodfellow *et al.*, "Challenges in Representation Learning: A Report on Three Machine Learning Contests," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8228 LNCS, no. PART 3, pp. 117–124, 2013, doi: 10.1007/978-3-642-42051-1_16.

[13] L. Pham, T. H. Vu, and T. A. Tran, "Facial expression recognition using residual masking network," *Proceedings - International Conference on Pattern Recognition*, pp. 4513–4519, 2020, doi: 10.1109/ICPR48806.2021.9411919.

[14] Y. Khaireddin and Z. Chen, "Facial Emotion Recognition: State of the Art Performance on FER2013," May 2021, Accessed: Sep. 19, 2021. [Online]. Available: https://arxiv.org/abs/2105.03588v1

[15] C. Pramerdorfer and M. Kampel, "Facial Expression Recognition using Convolutional Neural Networks: State of the Art," Jan. 2016.

[16] I. G. P. Kusuma Negara, J. Jonathan, and A. Lim, "Emotion Recognition on FER-2013 Face Images Using Fine-Tuned VGG-16," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, pp. 315–322, Jan. 2020, doi: 10.25046/aj050638.

[17] H.-D. Nguyen, S. Yeom, G.-S. Lee, H.-J. Yang, I. Na, and S. H. Kim, "Facial Emotion Recognition Using an Ensemble of Multi-Level Convolutional Neural Networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, Jan. 2018, doi: 10.1142/S0218001419400159.

[18] H. M. D. Kabir *et al.*, "SpinalNet: Deep Neural Network with Gradual Input," Jul. 2020, Accessed: Sep. 19, 2021. [Online]. Available: https://arxiv.org/abs/2007.03347v2

[19] P. Chopra, "ProgressiveSpinalNet architecture for FC layers," Mar. 2021, Accessed: Sep. 19, 2021. [Online]. Available: https://arxiv.org/abs/2103.11373v1

[20] D. Matsumoto and P. Ekman, "The relationship among expressions, labels, and descriptions of contempt," *Journal of Personality and Social Psychology*, vol. 87, no. 4, pp. 529–540, Oct. 2004, doi: 10.1037/0022-3514.87.4.529.

[21] P. Ekman and E. L. Rosenberg, "What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)," *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)*, pp. 1–672, Mar. 2012, doi: 10.1093/ACPROF:OSO/9780195179644.001.0001.

[22] S. Minaee and A. Abdolrashidi, "Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network." Jan. 2019.

[23] K. O'Nell, R. Saxe, and S. Anzellotti, "Recognition of identity and expressions as integrated processes." Apr. 2019. doi: 10.31234/osf.io/9c2e5.

[24] C. Li, N. Ma, and Y. Deng, "Multi-Network Fusion Based on CNN for Facial Expression Recognition," Jan. 2018. doi: 10.2991/csece-18.2018.35.

[25] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Sep. 23, 2021. [Online]. Available: https://arxiv.org/abs/1409.1556v6

[26] C. Szegedy *et al.*, "Going Deeper with Convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1–9, Sep. 2014, Accessed: Sep. 23, 2021. [Online]. Available: https://arxiv.org/abs/1409.4842v1

[27] M. Tan and Q. v. Le, "EfficientNetV2: Smaller Models and Faster Training," Apr. 2021, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/2104.00298v3

[28] M. Tan and Q. v. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, Accessed: Jan. 23, 2021. [Online]. Available: http://arxiv.org/abs/1905.11946

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Jan. 2018, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/1801.04381v4

[30] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 2011–2023, Sep. 2017, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/1709.01507v4

[31] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 1800–1807, Oct. 2016, Accessed: Oct. 07, 2021. [Online]. Available: https://arxiv.org/abs/1610.02357v3

[32] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Oct. 2020, Accessed: Oct. 08, 2021. [Online]. Available: https://arxiv.org/abs/2010.11929v2

[33] P.-L. Carrier and A. Courville, "Challenges in Representation Learning: Facial Expression Recognition Challenge," Apr. 13, 2013. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data (accessed Jan. 24, 2021).

[34] X. Wang *et al.*, "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11133 LNCS, pp. 63–79, Sep. 2018, Accessed: Oct. 23, 2021. [Online]. Available: https://arxiv.org/abs/1809.00219v2

[35] "GitHub - lessw2020/Ranger-Deep-Learning-Optimizer: Ranger - a synergistic optimizer using RAdam (Rectified Adam), Gradient Centralization and LookAhead in one codebase." https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer (accessed Oct. 17, 2021).

[36] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead Optimizer: k steps forward, 1 step back," *Advances in Neural Information Processing Systems*, vol. 32, Jul. 2019, Accessed: Oct. 17, 2021. [Online]. Available: https://arxiv.org/abs/1907.08610v1

[37] D. Misra, "Mish: A Self Regularized Non-Monotonic Activation Function," Aug. 2019, Accessed: Oct. 17, 2021. [Online]. Available: https://arxiv.org/abs/1908.08681v3

[38] "GitHub - huggingface/transformers: Transformers: State-of-the-art Natural Language Processing for Pytorch, TensorFlow, and JAX." https://github.com/huggingface/transformers (accessed Oct. 17, 2021).

[39] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/J.IPM.2009.03.002.

[40] L. Wright and N. Demeure, "Ranger21: a synergistic deep learning optimizer," Jun. 2021, Accessed: Oct. 17, 2021. [Online]. Available: https://arxiv.org/abs/2106.13731v2