Project 2: Gesture Recognition

Chayapol Hongsrimuang

1 Introduction

This project concerns the gesture recognition of hands using a convolutional neural network (or CNN, for short) This type of algorithm is designed for classification tasks in images such as object recognition, or in the case of the project, hand gesture recognition. The neural network has three main layers that are used: [Keita, 2023]

- Convolutional layer - this is where several filters are applied to recognise a pattern from the image
- Pooling layer - this layer would pull the significant features from the convolutional layer to a pooling result. This reduces the memory used when training. Common functions for pooling layer include: max pooling (maximum of a XxY filter) and average pooling (average of a XxY filter)
- Fully-connected layers - this set of layers are used to flatten the results from the pooling layer to a one-dimensional matrix. Activation functions such as ReLU are then applied, with softmax prediction layer as well.

The predictions from the last layers are then used to determine the classification of the image. The higher the probability, the more likely the class belongs to that image, with the highest one being the class the image belongs to, based on the predictions.

Not only that convolutional neural networks are explored in this project, transfer learning, based on convolutional neural networks are also explored in this project. This is where a pre-trained model is re-used to predict another set of data. For example, if a model is trained for identifying animals, that model can then be reused to identify gestures. [Sharma, 2023]

There are many pre-trained models that are available, such as Xception, Inception v3, MobileNet, etc. For this project, VGG16 is used as the pre-trained model for transfer learning. This model makes use of around 11-19 weighted layers, with a 224x224 RGB (Red, Green, Blue colour space) image input. The pooling in their case is a 2x2 window with a stride of 2. A filter of 3x3 is also used for their convolutional layers. [Simonyan and Zisserman, 2014]

1.1 Dataset

The dataset used is a HaGRID. (HAnd Gesture Recognition Image Dataset) This dataset is a hand gesture recognition dataset, containing 18 different gestures that can be identified using this dataset. These gestures (or classes) are as follows: call, dislike, first, four, like, mute, ok, one, palm, peace, peace inv, rock, stop, stop inv, three, three2, two up, two up inv. [Kapitanov et al., 2024]

The dataset contains approximately 770 GB worth of images, including more than 550 thousand images to be used in image recognition models, such as ones created for the project. [Kapitanov et al., 2024] However, only 125,912 images are used (totalling to the size of under 4GB) due to the enormous size of the dataset. The images are also further reduced to 512x512 size for the project.

1.2 Project Objectives

This project's main objective is to create models using convolutional neural networks as well as transfer learning from a pre-trained model, VGG16. A number of models are created with different configurations, including manually-defined CNNs and a transfer learning model.

The models would then be compared with each other to find out the best model to be used to identify the project's own pictures. These are provided in the Appendix. They are also under the /testing_own folder, with folder names corresponding to the class names they belong to.

2 Methodology

For each model, the dataset is loaded depending on the type of model that is used. The following configurations are done to load the images:

- Batch Size - 125,912
- Image Size - 512x512
- Seed Number - 388741 (corresponding to the student ID)

The dataset is loaded via greyscale or RGB. This depends on the model that is used, e.g. if the model is greyscale, the dataset loaded would be greyscale, and vice versa. The dataset is then split further into three separate data sets:

- Training - 70%
- Validation - 10%
- Test - 20%

The training and validation data sets are used in training the models. The test data set is used to evaluate the model after the models are trained.

The datasets themselves, as described earlier in the HaGRID section, contained 18 different classes, each in their respective folders, with classes as the names of the folders. The classes are indicated in the earlier section.

The data sets' image size remained unchanged for training the manually-created models. However, the image size used for transfer learning is changed from 512x512 to 224x224, due to the memory cost to use the pre-trained model against the images. The 224x224 image size is the default value for the pre-trained model of VGG16. [Simonyan and Zisserman, 2014]

Each of the model declared also have a common value of 50 epochs/iterations. However, the number of epochs may not be detrimental due to early stopping mechanism.

Early stopping is implemented for each of the model, to prevent overfitting of the models. This is based on the loss function (how well the model is done against the data set) of the validation set, with patience of 3, where the early stopping function would wait for 3 epochs, before the model stops training if there are no improvements in the loss function.

3 Models

There are a total of 5 different models that are declared for this project. After each of the convolutional layer, a pooling layer is inserted before the next convolutional layer (or fully-connected layers). The filter size is also the same of 3x3. (defined via kernel sizes) These models are indicated in Table 1.

Each of the manually created CNNs contains a stride of 2. This dictates the movement of the kernel, by which the movement is by two pixels at a time. This produces a smaller image size to be passed on, but comes

Configurations for each convolutional neural network					
Model Number and Name	Colour Space	Stride	Convolutional Layers (with filters number)	Kernel Size	Pooling Type
1 (Greyscale Simple)	greyscale	2	2 (16, 32)	3	Maximum
2 (Greyscale Deep)	greyscale	2	8 (12, 18, 24, 32, 48, 96, 128, 192)	3	Maximum
3 (RGB Deep)	RGB	2	8 (12, 18, 24, 32, 48, 96, 128, 192)	3	Maximum
4 (RGB Deep w/ Dropout)	RGB	2	8 (12, 18, 24, 32, 48, 96, 128, 192)	3	Maximum
5 (RGB Transfer Learning)	RGB	1	None (pre-trained)	-	Average

Table 1: Model configurations

at a cost of fewer features being passed through. However, this is another method to save memory when it comes to training a convolutional neural network.

For model 4, a dropout layer is also defined after every pooling layer, except the last layer. The dropout percentage is 20% for that layer. This mechanism is to reduce overfitting for the model, with 20% of the nodes being dropped randomly for each layer. This also can make the model more generalised. [Yadav, 2023]

For model 5, the RGB colour space is used instead of greyscale, this is due to the amount of features that the model can be passed into, helping the accuracy to be a bit higher. (x3 the image size vs x1 the image size, for each pixel) The comparison between each colour space would be more magnified in the Results section, in terms of accuracy and loss.

4 Results

When testing with the testing set from HaGRID, the results from each of the model is described in Table 2.

Results from testing dataset of each model					
Model Number and Name	Accuracy (%)	Loss	Training Time	Epochs	Speed (epoch/time)
1 (Greyscale Simple)	15.9%	2.705	22.3 mins	11	0.49
2 (Greyscale Deep)	80.2%	0.664	17.2 mins	10	0.58
3 (RGB Deep)	84.0%	0.526	22.8 mins	9	0.39
4 (RGB Deep w/ Dropout)	76.4%	0.769	38.1 mins	12	0.31
5 (RGB Transfer Learning)	65.0%	1.193	64.1 mins	7	0.11

Table 2: Model results from testing set

The training time is based on the sum of time indicated in each epoch. For the fifth model, there is an instance of an epoch taking a lot of time to train. This was due to the computer being in sleep mode at the time. In this case, the average of the other epochs is used for that epoch, and then calculated in total again.

Based on this table, the best model is chosen by the best accuracy percentage, which is model number 3 (RGB Deep) The training time for the model is also reasonable as well, with the model taking 22.8 minutes to train, which is comparable to the simple greyscale model. However, the training time is understandably higher than a deep greyscale model, with more features. The dropout model seems to not perform well in training time, due to the amount of epochs themselves and fluctuating validation error loss. The speed of training is also not great as well, but better than with the dropout mechanism. However, the accuracy makes the model preferable to be used.

When choosing model 3, the loss between the training set and the validation set is roughly the same as well, with a difference of around 0.2 in the last epoch. This is indicated in Figure 1.

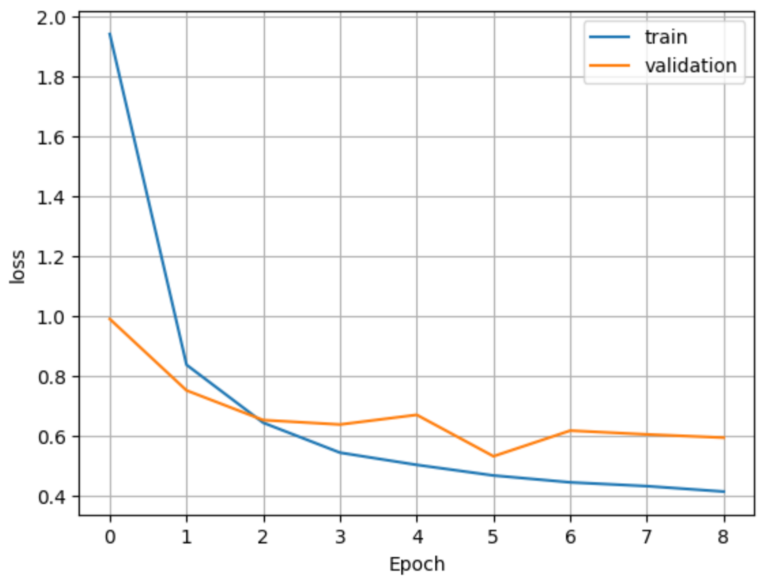

Figure 1: The loss function graph for model 3 - RGB Deep (training vs validation)

However, this may indicate some overfitting for the model itself, as the validation loss is greater than the training loss. This is also the same for the testing set's loss, but less difference. When comparing this to model 4 with similar configurations, but with a dropout mechanism, there is less overfitting. However, the accuracy is dropped to around 76% (from testing set) with this mechanism. The loss training graph for model 4 is shown in Figure 2.

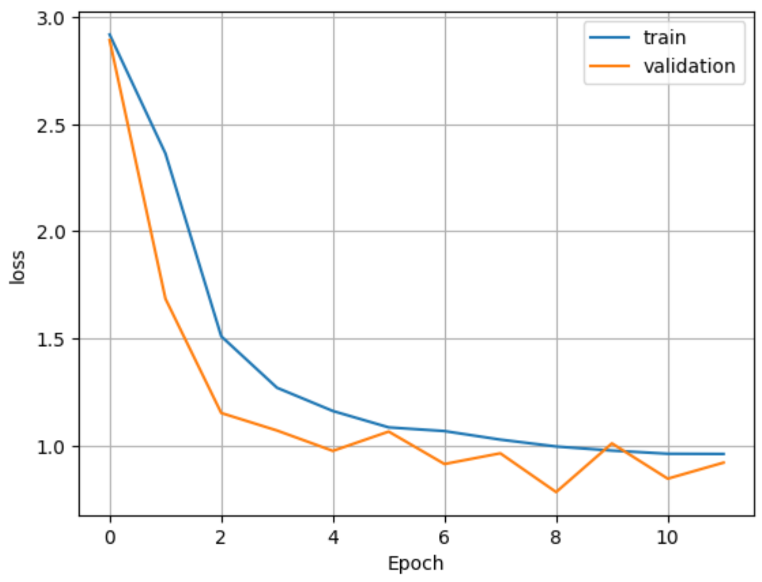

Figure 2: The loss function graph for model 4 - RGB Deep with Dropout (training vs validation)

Otherwise, despite the overfitting nature of model 3, model 3 is still the best model that can be used to accurately predict gestures from an external testing data. (not from HaGRID set)

4.1 Best model testing

With model 3 being chosen as the best model for accuracy, testing is done on the set of data created for this project. (external images taken for the project other than the HaGRID set) The results are defined in Table 3.

Predictions of external testing set from the best model		
Picture Name	Prediction	Prediction Percentage
ok	ok	13.8%
one	one	13.7%
peace	peace	13.5%
stop	stop	13.6%

Table 3: Model predictions for external testing set

Table 3 proves that the best model of model 3 can determine all four pictures correctly, with a high percentage of around 13%. The rest of the predictions for other classes were around 5%.

5 Conclusion

This project tested different convolutional neural network models to find out what configuration is the best to identify a set of gesture images. The results turned out to be that the model with high amount of convolutional and pooling layers (with progressively increasing amount of filters) as well as high amount of features (from the RGB colour space) would produce a high amount of accuracy. This makes it the best model in terms of accuracy. However, there can be some problems when it comes to overfitting of the model, with the training error being lower than the validation error.

Nevertheless, there are some techniques that are explored in reducing overfitting. These include early stopping and dropout. However, the dropout technique produced less accuracy when the same layer structure is used from the best model. With this, the model with dropout may require additional adjustments, including deeper models or other techniques such as data augmentation (where the data set are adjusted to produce more data via rotating, reflection, etc.) and regularisation. Data augmentation is not included in making the models as they can produce errors when working with the models in this project.

This project proved that deeper models tend to produce high accuracy when it comes to classification tasks such as gesture recognition. Features are also detrimental as well, with higher image size and colour space, producing better results. This is proven by comparing the greyscale models vs the RGB models, as well as the RGB models vs a transfer learning model (with the transfer learning model having lower image size)

References

[Kapitanov et al., 2024] Kapitanov, A., Kvanchiani, K., Nagaev, A., Kraynov, R., and Makhliarchuk, A. (2024). Hagrid–hand gesture recognition image dataset. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4572–4581.

[Keita, 2023] Keita, Z. (2023). An introduction to convolutional neural networks: A comprehensive guide to cnns in deep learning.

[Sharma, 2023] Sharma, P. (2023). Understanding transfer learning for deep learning.

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

[Yadav, 2023] Yadav, H. (2023). Dropout in neural networks.

Appendix

A Images used for testing the best model

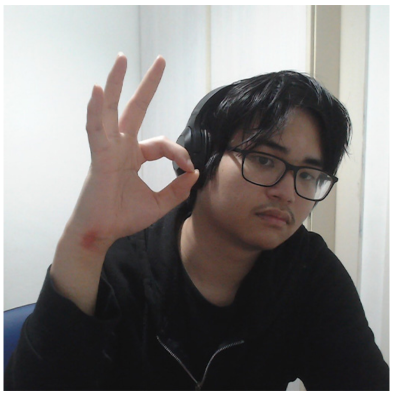
Figure 3: Class - ok

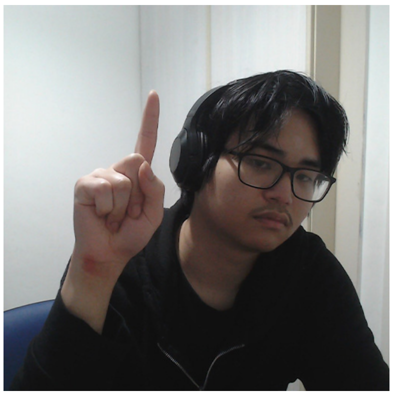
Figure 4: Class - one

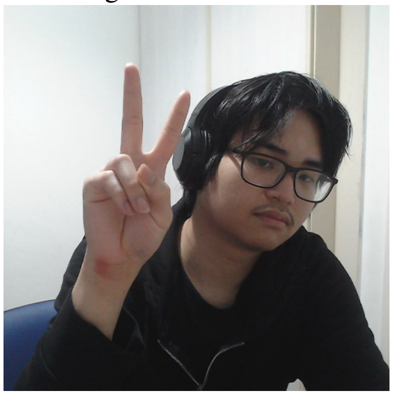
Figure 5: Class - peace

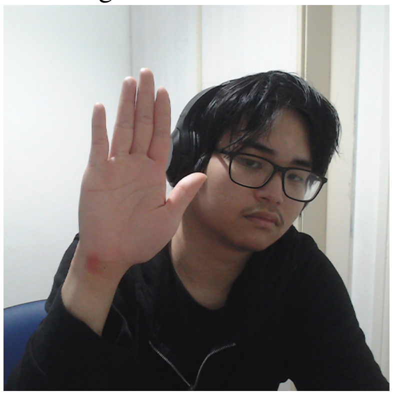
Figure 6: Class - stop