Machine Learning Final Project Report

110511010 楊育陞 110511067 葉哲伍

1 Introduction

In this project, we designed and trained a machine learning model to do a binary classification task. The classification is to predict whether a person in image is an adult or a child. We designed a custom lightweight CNN model to achieve this task.

2 Train of Thought

2.1 Observations on the dataset

- The dataset is composed of about four hundred images of adults and children separately, which is not a large dataset.
- Two classes are balanced.
- For some data samples as Figure 1, we can observe that:
  - The people in the images are in different poses, sizes, and backgrounds.
  - The images are in different lighting conditions and color tones.

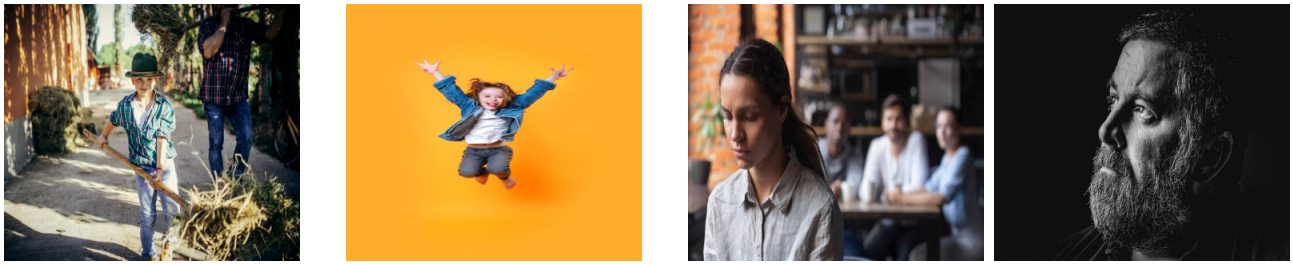

Figure 1: data samples

2.2 Methods explored

We explored the following methods to achieve the classification task:

- Binary logistic regression.
- Support vector machine.
- K-nearest neighbors.
- Neural networks.

The experiments results of the first three methods are as Table 1

Analyzing the results, we found that the performance of the three methods is not satisfactory. The accuracy is not high enough, and the models are not robust. The reason is that the three methods can't extract features. Manually extracting features is time consuming and not efficient. So we decided to use neural networks to automatically extract features.

	Logistic Regression	C-SVM	KNN
Accuracy	52.5%	58.3%	60.0%

Table 1: Accuracy on non neural network models

2.3 Neural network building

For the image processing task, CNN and transformer are the most popular neural network architectures. After surveying the state-of-the-art models on image classification, we decided to build a custom lightweight CNN model to achieve the classification task. The reasons are as follows:

- The dataset is not large, so we don't need a very deep model.
- Our resources are limited, so we can't train a very large model.
- Transformer models usually require a large dataset and a lot of computation resources, which is not suitable for this task and our resources.

Other than custom models, we also considered fine-tuning pre-trained models or using face detection models to aid the classification task. But we found that all these methods are not suitable for this task. The reason is that these models need relatively high computational resources, which our computation resources can't handle.

3 Method

3.1 Data Augmentation

Considering the observations above, we applied the following data augmentation techniques to the dataset:

For people in the images are in different poses and sizes:

- Random Horizontal Flip: We flipped the images horizontally with a probability. This technique can help the model to learn because the people horizontally flipped are still adults or children.
- Random Affine: We applied random affine transformations to the images, which is to rotate, scale, and translate the images. This technique can help the model to learn because the people in the images are in different poses and sizes.
- Random Perspective: We applied random perspective transformations to the images, which is to warp the images. This technique is similar to random affine.

For images are in different lighting conditions and color tones:

- Color Jitter: We applied color jitter to the images, which is to change the brightness, contrast, saturation, and hue of the images. This technique can help the model to learn because the people in the images are in different lighting conditions and color tones.
- Random Grayscale: We converted the images to grayscale with a probability. This technique can help the model to learn because some images are in grayscale.

Figure 2: Random Perspecitve

Figure 3: Color Jitter

3.2 Feature Extraction and Classification

We split the features extraction and classification into three parts as Figure 4.

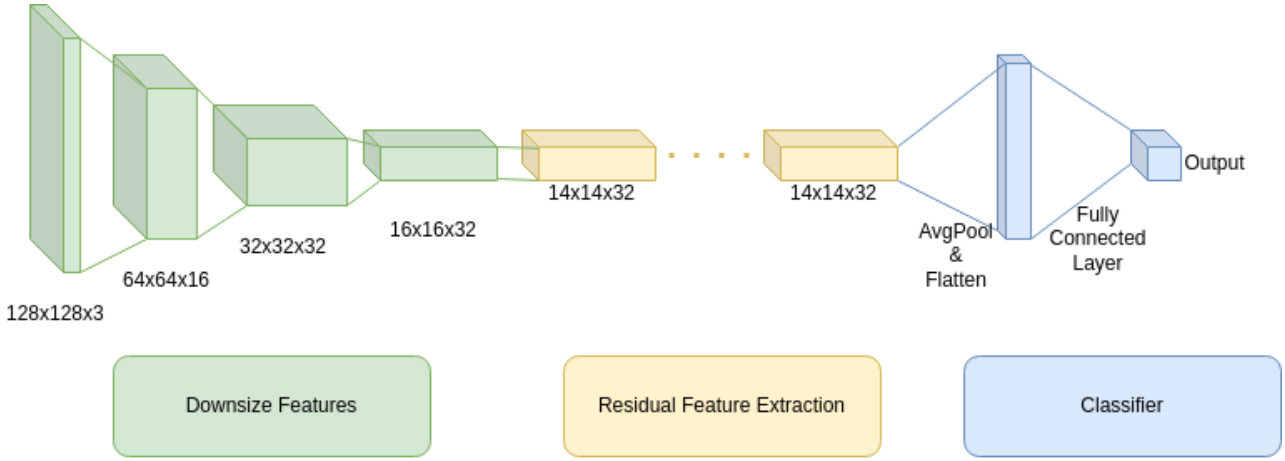

Figure 4: Feature Extraction and Classification

3.2.1 Downsize features

In this part, we extracted and downsize the features of the images to reduce the computation cost while keeping the important information. We used the max pooling layer to downsize the features.

3.2.2 Residual feature extraction

For this part, we extracted and remapped the features of the images using relatively deep residual blocks. This part highly affects the complexity of the model.

3.2.3 Classifier

The last part is the classifier, which is a fully connected layer. First we use average pooling to reduce the dimension of the features. Then we use a fully connected layer to generate the output.

After splitting the model into these three parts, we adjusted the scale of the model by modifying the depth and width of these parts. Also we first use simple structures to build these parts. Then we changed the structures to more complex ones to improve the performance after determining the scale of the model.

3.3 Model Structure

Our model structure is as following figure. We will discuss the details of the model structure in the order of the three parts.

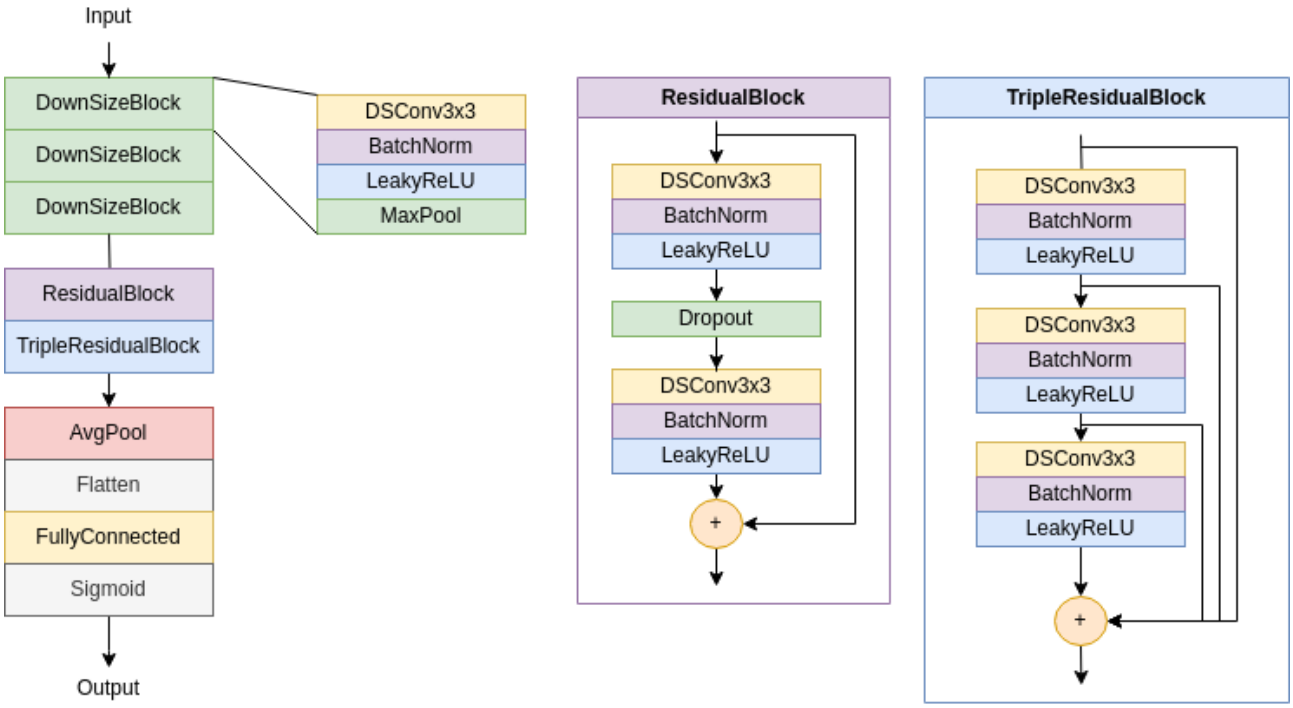

Figure 5: Model Structure

3.3.1 Downsize features

In this part, we used three 'DownSizeBlock's. The block is composed of a depthwise separable convolution layer, a batch normalization layer, a leaky ReLU activation function, and a max pooling layer.

- Depthwise separable convolution: This is proposed in the MobileNet model. It is a lightweight convolution layer that can reduce the computation cost while keeping the important information.
- Batch normalization: This normalizes the features among the batches. It can help the model to converge faster and generalize better. Also it can lessen the effect of the internal covariate shift.

- Leaky ReLU: This is a variant of the ReLU activation function. It keeps the negative values of the input with a small slope. This can keep the information of the negative values and help the model to learn.
- Max pooling: This downsizes the features by taking the maximum value of the input. This can reduce the computation cost while keeping some important information.

3.3.2 Residual feature extraction

For this part, we used a 'ResidualBlock' and a 'TripleResidualBlock'.

- ResidualBlock: This is a simple residual block. As figure 5 shows, it is composed of two blocks of depthwise separable convolution layers, batch normalization layers, and leaky ReLU activation functions. After the first block, we added a dropout layer. Also we added a skip connection to the output of the first block from the input. This can help the model to learn the residual features.
- TripleResidualBlock: This is a complex residual block. It is composed of three residual blocks. All the outputs of the three blocks and the input are added together to the output. Each block builds on the previous block, which enhances the data progression.

3.3.3 Classifier

The classifier is composed of an average pooling layer, a fully connected layer, and a sigmoid activation function.

- Average pooling: This reduces the dimension of the features by taking the average value of the input. This can reduce the computation cost while keeping some important information.
- Fully connected layer: After the average pooling layer, we flatten the output and use a fully connected layer to generate the output. This can help the model to learn the features and classify the images.
- Sigmoid activation function: This is used to generate the output and calculate the loss.

3.4 Training

- Loss function: We used the binary cross entropy loss function. This is suitable for binary classification tasks.
- Optimizer: We used the Adam optimizer. This is a popular optimizer that can adaptively adjust the learning rate.
- Batch size: We used a batch size of 64. This is suitable for our computation resources and the dataset size.

4 Result

After times of experiments, we have achieved the final model of accuracy 78% average (last 100 epochs) and 85% best (last 100 epochs). The model is with number of parameters 9.823k and FLOPs 17.63M. The learning curve is as following figure.

4.1 Accuracy and Learning Curve

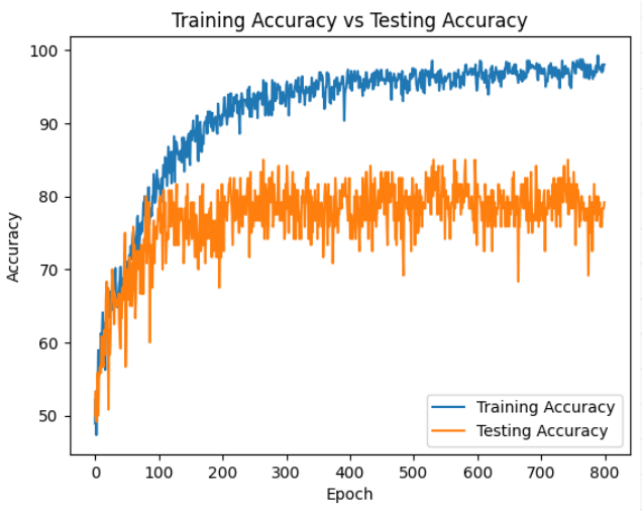

Figure 6: Learning Curve

- Training Accuracy converges at about 97%
- The Highest Testing Accuracy is about 85%
- The Average Testing Accuracy is about 78% average after 200 epoch

5 Ablation Study

5.1 Data augmentation

The result is as following figure.

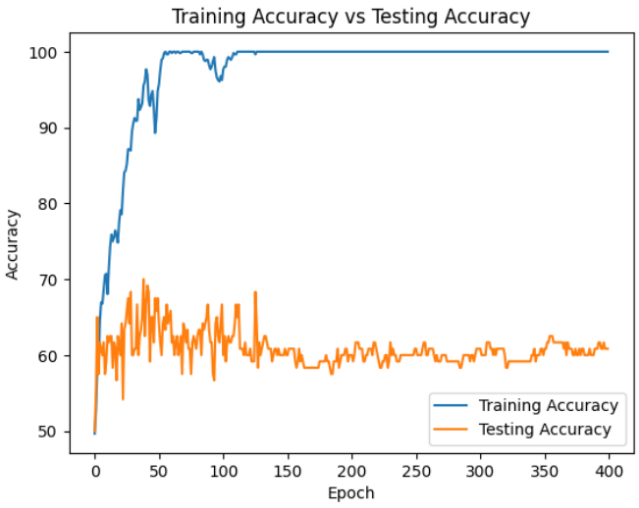

Figure 7: Learning Curve w/o data augmentation

Without data augmentation, the model converges faster but highly overfits the training data, which means the model generalizes poorly.

5.2 Activation function

We tried the ReLU activation function, the leaky ReLU activation function, and the PReLU activation function. The result is as following:

	ReLU	Leaky ReLU	PReLU
Accuracy	76.5%	78.0%	78.5%

The PReLU activation function performs the best. The reason is that the PReLU activation function can learn the slope of the negative values, which can help the model to learn the features. But the PReLU activation function increases the number of parameters and computation cost with little improvement in performance. So we decided to use the leaky ReLU activation function.

5.3 Batch normalization

We tried the model with, without batch normalization, and with group normalization. The result is as following:

	Without BN	With BN	With GN
Accuracy	70.0%	78.0%	76.7%

The model with batch normalization performs the best. The reason is that the batch normalization can normalize the features among the batches, which can help the model to converge faster and generalize better. The group normalization performs worse after experiments.

5.4 Residual connection

Without residual connection, the model performs worse(73.5%) after experiments.

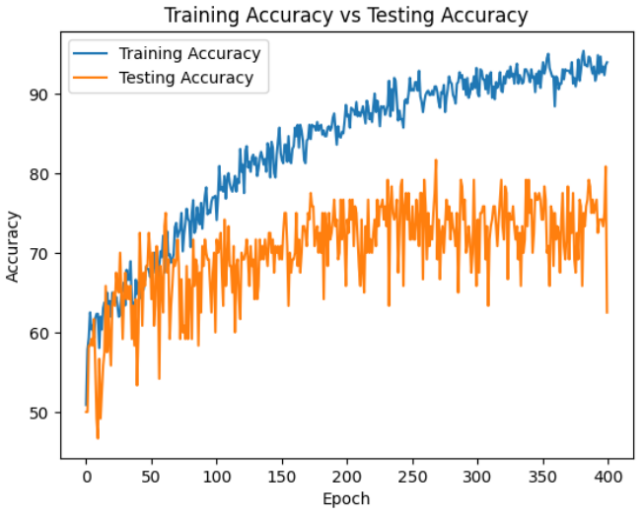

Figure 8: Learning Curve w/o residual connection

5.5 Residual block vs triple residual block

We also tried the following combinations of residual blocks.

- Residual block + Residual block
- Residual block + Triple residual block
- Triple residual block + Residual block
- Triple residual block + Triple residual block

The model with residual block + triple residual block performs the best. So we decided to use this structure.

5.6 Depthwise separable convolution

We also tried the model with normal convolution layers. The result is similar to the model with depthwise separable convolution layers, but has much more parameters and computation cost as following table:

	Normal Conv	Depthwise Separable Conv
Parameters	61.345k	9.8k
FLOPs	72.376M	17.6M
Accuracy	78.5%	78.0%

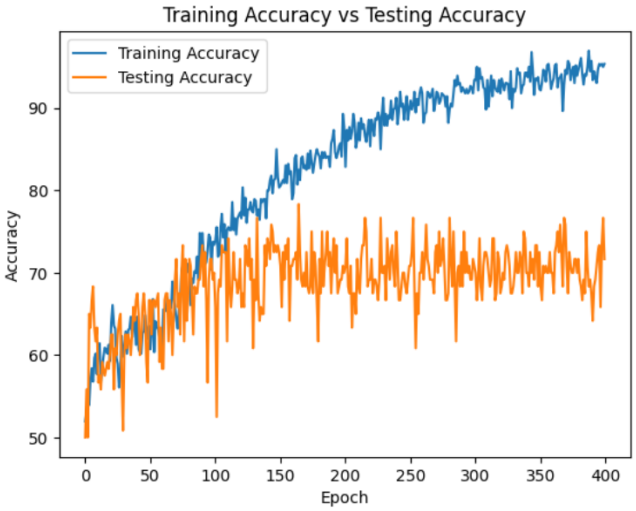

Figure 9: Learning Curve w/ original convolution

6 Conclusion

In this project, we designed and trained a custom lightweight CNN model. We applied data augmentation techniques to the dataset. We split the model into three parts: downsize features, residual feature extraction, and classifier. We used the leaky ReLU activation function, the batch normalization layer, the residual connection, and the depthwise separable convolution layer. After experiments, we achieved the final model of accuracy 78% average (last 100 epochs) and 85% best (last 100 epochs) on testing dataset. The model is with number of parameters 9.823k and FLOPs 17.63M.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition.

[2] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.