Neural Network Classitication

Project Machine Learning

Dao Ton Son

Contents

1 Introduction

1.1 Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1.2 Neural network for classifiication problem

Neural network classification is a powerful technique that has revolutionized the field of image analysis and recognition. With the rise of deep learning and convolutional neural networks (CNNs), trained model are now capable of accurately classify pictures accross a wide range of categories. This breakthrough technology has found applications in various domains, including medical imaging, autonomous driving, facial recognition, and more.

The ability to classify pictures using neural networks holds immense potential for understanding and interpreting visual data. Neural networks can learn complex patterns and relationships within the data, enabling them to make accurate predictions on unseen images. One of the key advantages of neural network classification is its ability to automatically extract meaningful features from raw images. Instead of relying on handcrafted features, such as color histograms or texture descriptors, the neural network learns to recognize important and distinguished features directly from the pixel values. Convolutional neural networks

(CNNs), in particular, excel at this task by utilizing convolutional and pooling layers to capture local patterns and hierarchical representations within the images.

In this project, I will attempt to build a convolutional neural network model to classify pictures of dogs and cats. In the first part, I will introduce the data set and the prepossessing for the data. In the second part, I build my neural network along with a baseline model that follows visual geometry group (vgg) model [1]. Next part demonstrate how I train the data and the setup and I also show how I assess different models to find the best one and alternatives model regarding the structure, hyper parameter,etc. Finally, using 5-fold cross-validation, I compare the risk estimates of my baseline model and the one that I assemble.

2 Data and Preprocessing

The data set is compiled of 25 thousands pictures of both cats and dogs. There are corrupt files, which are identified and deleted, among the pictures Firstly, pictures of different sizes are transformed to RGB pixel value and scale down to 100 by 50 pixels. Data with higher values in height and width does not improve enough accuracy in proportion to its run-time to train the model. Small value in height and width decrease significantly the performance of the classification model. Not only that, since there is max pooling layer, having less pixels means that in the later stage of the model, the number of meaningful values will not be sufficient for an effective selection of features in max pooling layer. Secondly, pictures of dogs and cats are combined and splitted into test (20%) and train (80%) set. Validation set is 20% of the train set. The value in each pixels is normalized to a range from 0 to 1. Finally, the images in the training set will be augmented by shifting 10% horizontally and vertically.

3 Convolutional Network Configuration

3.1 Baseline model

A baseline model in neural networks is a simple and straightforward model that serves as a starting point or reference for comparison. In another word, the purpose of the baseline model is to establish a benchmark performance against which the performance of more complex or sophisticated models can be evaluated. The model that is chosen to be my benchmark model is vgg model proposed by Karen Simonyan and Andrew Zisserman [1]. The architecture of the model is easy to understand and implement. The structure involves stacking convolutional layers with small 3×3 filters with stride 1 followed by a max pooling layer (2x2) with stride 2. Both layers are combines to form a block of vgg and these blocks can be put one after another where the number of filters in the convolutional layer double after every block starting from 32 (e.g 32, 64,128, etc.). In order to ensure the height and width shapes of the output feature maps matches the inputs, padding is used in the convolutional layers. Each hidden layer and the dense layer with 128 filters in the output layer are equipped with the rectification (ReLU [2]) non-linearity. ReLU function is defined as

$$f(x) = max(0, x)$$

where x is the input to the activation function. The function will return 0 if the input is negative and return the input if it is positive. For this project, vgg model will be comprised of 3 vgg-blocks and have the following structure (show structure of the model).Then the hidden layer is connected to a dense layer with 128 filters. Finally, the dense layer is connected to the last output layer with sigmoid activation function to determine the probability of the input picture to be a dog or a cat.

3.2 My Model

My model is an adaptation of the vgg model. I take advantage of the same structure of a vgg block in the hidden layer where each block of mine is incorporated with one convolutional layer with a small receptive field 3x3 with stride 1 that is followed by a layer of max pooling(2x2) with stride 2. I have the same ReLU activation fuction in each layer in the hidden layer and output layer accept the last layer of output layer which uses sigmoid activation function. However, I have an addition vgg block with 512 nodes for additional feature extraction enhancement. Despite the minimal changes in the structures between my baseline model and my model, the discussion later will explain how the difference of the two models have important impacts on the their prediction performance and why I only have one more vgg block compare to my baseline model.

4 Classification Framework

In the previous section, I demonstrate the features of the models' configuration. In this section, I will present the process of training and evaluating the models.

4.1 Training

For the baseline mode, based on back-propagation [3], the training process uses stochastic gradient descent to optimizing the multinomial logistic regression target with learning rate equals to 10^{-3} and momentum set to 0.9. At the dense layer, drop out rate is set to 0.5. The drop out rate relate to the number of random nodes in each layer will be removed while training the model. Drop out ratio prevent the neural network from relying on certain nodes in the network to make prediction. The batch size is set to 64. One more important factor contribute to a good training process is weight initialization because bad initialization can lead to diminishing or exploding gradient preventing the learning procedure to minimizing the loss function. To avoid this problem, in every layer of the network, I set the weight to follow glorot uniform distribution [4]. The Glorot distribution ensures that the variance of the weights for each node is approximately equal to the variance of the inputs.

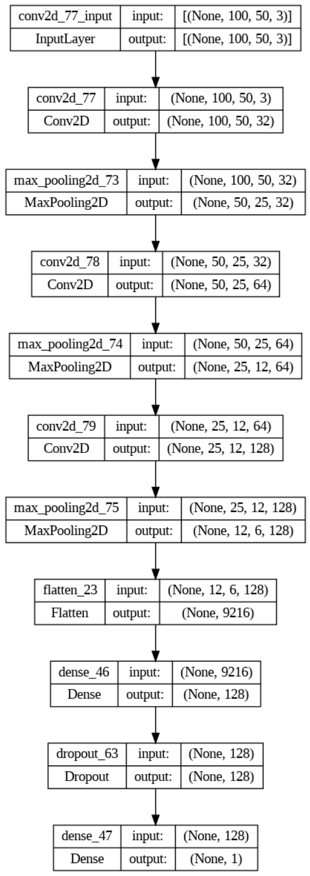
For my model, I also train with size 64 but instead of using stochastic gradient descent(SGD) to perform parameters update for each iteration, I deploy adaptive moment estimation (adam) [5] with exponential learning rate decay.

$$\text{decay learning rate} = \text{learning rate} \cdot \text{decay rate}^{\left(\frac{\text{global step}}{\text{decay steps}}\right)}$$

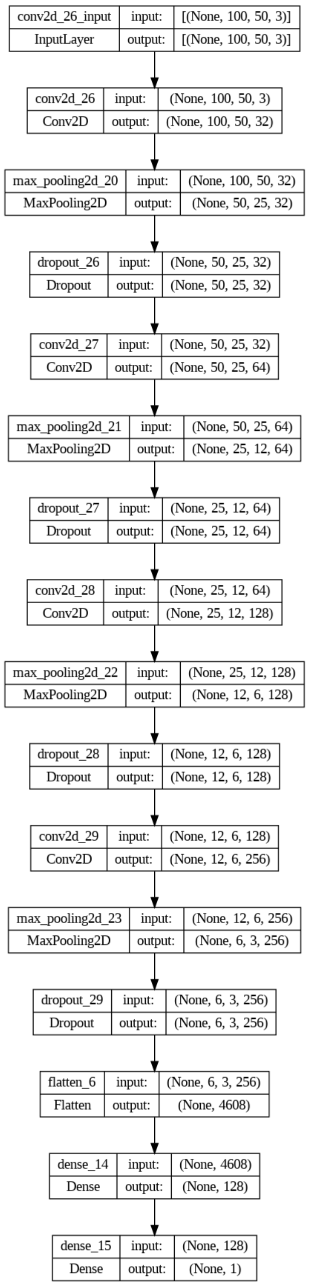
I set the learning rate is 10^{-3}, decay rate is 0.95, global step is 8000 and the decay step is 10^4. I only regularlize by having a drop out rate of 0.2 for each layer in the hidden layer and not in the dense layer like in the base model. By applying this type learning rate, the learning rate is gradually decrease during the training process and the model will update the parameters converging the loss function easier. This makes the output more stable during validation process compare to the baseline model.

Figure 1 show the complete structure of the two models number of epoch for each model

I calculate loss function using binary cross entropy which is a popular and effective method for binary classification problem [6]. For both model, I train for 30 epoch.


conv2d_77_input | input: | [(None, 100, 50, 3)]
InputLayer | output: | [(None, 100, 50, 3)]
conv2d_77 | input: | (None, 100, 50, 3)
Conv2D | output: | (None, 100, 50, 32)
max_pooling2d_73 | input: | (None, 100, 50, 32)
MaxPooling2D | output: | (None, 50, 25, 32)
conv2d_78 | input: | (None, 50, 25, 32)
Conv2D | output: | (None, 50, 25, 64)
max_pooling2d_74 | input: | (None, 50, 25, 64)
MaxPooling2D | output: | (None, 25, 12, 64)
conv2d_79 | input: | (None, 25, 12, 64)
Conv2D | output: | (None, 25, 12, 128)
max_pooling2d_75 | input: | (None, 25, 12, 128)
MaxPooling2D | output: | (None, 12, 6, 128)
flatten_23 | input: | (None, 12, 6, 128)
Flatten | output: | (None, 9216)
dense_46 | input: | (None, 9216)
Dense | output: | (None, 128)
dropout_63 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_47 | input: | (None, 128)
Dense | output: | (None, 1)

(a) Baseline model architect


conv2d_26_input | input: | [(None, 100, 50, 3)]
InputLayer | output: | [(None, 100, 50, 3)]
conv2d_26 | input: | (None, 100, 50, 3)
Conv2D | output: | (None, 100, 50, 32)
max_pooling2d_20 | input: | (None, 100, 50, 32)
MaxPooling2D | output: | (None, 50, 25, 32)
dropout_26 | input: | (None, 50, 25, 32)
Dropout | output: | (None, 50, 25, 32)
conv2d_27 | input: | (None, 50, 25, 32)
Conv2D | output: | (None, 50, 25, 64)
max_pooling2d_21 | input: | (None, 50, 25, 64)
MaxPooling2D | output: | (None, 25, 12, 64)
dropout_27 | input: | (None, 25, 12, 64)
Dropout | output: | (None, 25, 12, 64)
conv2d_28 | input: | (None, 25, 12, 64)
Conv2D | output: | (None, 25, 12, 128)
max_pooling2d_22 | input: | (None, 25, 12, 128)
MaxPooling2D | output: | (None, 12, 6, 128)
dropout_28 | input: | (None, 12, 6, 128)
Dropout | output: | (None, 12, 6, 128)
conv2d_29 | input: | (None, 12, 6, 128)
Conv2D | output: | (None, 12, 6, 256)
max_pooling2d_23 | input: | (None, 12, 6, 256)
MaxPooling2D | output: | (None, 6, 3, 256)
dropout_29 | input: | (None, 6, 3, 256)
Dropout | output: | (None, 6, 3, 256)
flatten_6 | input: | (None, 6, 3, 256)
Flatten | output: | (None, 4608)
dense_14 | input: | (None, 4608)
Dense | output: | (None, 128)
dense_15 | input: | (None, 128)
Dense | output: | (None, 1)

(b) My model architect

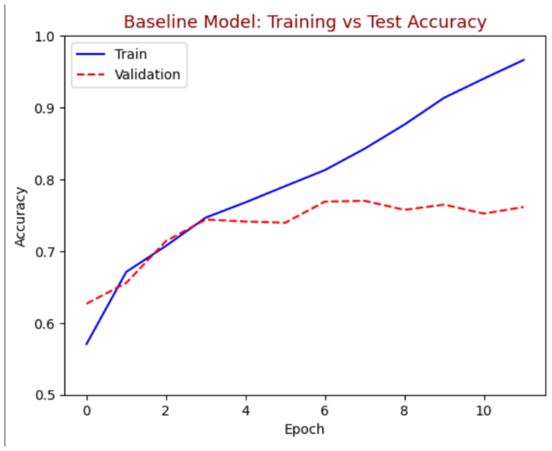
Figure 1: Model architect

5 Evaluation

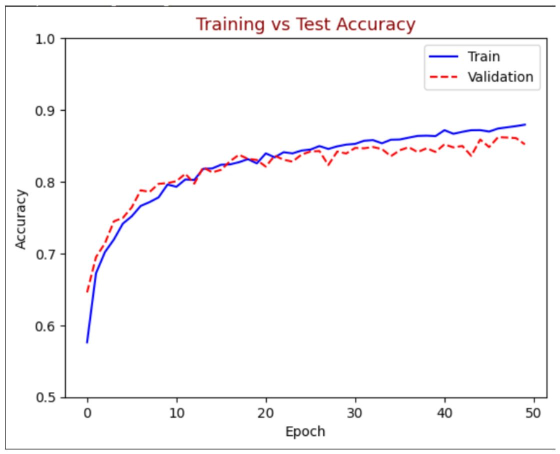
5.1 Model Assessment

This section demonstrate the process of how I select the best performing model. The model selection process is carried out by assessing test result and the graph generates by each model's training process showing the model's training and validation accuracy in classifying pictures of dogs and cats. I evaluate different models to find the best one. At first, I examine how the model perform after each epoch and how the value of loss function converge over time. For this one, after epoch 20, most model starts to stop significantly increase accuracy at 80% accuracy. My model converges at epoch 17 and reach 85% accuracy in the validation set, the highest accuracy compare to other models. The baseline model also stop improving much after 17 epoch. As you can see in figure 2, since epoch 10, the gap between training and validation accuracy increases as the training go on indicating overfitting in the model. The baseline model picks up noises during the training process in which it decreases bias and increases variance. When evaluating the models on the test set, baseline model got 82.14% accuracy and loss value is 0.5206 while my model got accuracy 84.12% and loss equals to 0.3557. This means my model reach an optimal state much faster and more precise compare to the benchmark model.

Different configurations for the convolutional network are considered during the assessment process.All other aspects of the models in figure 3 are kept the same except different features in each architecture. As it shows in the graph, train and validation accuracy stays close together during the while 30 epoch, proving the regularization works. In the begining, models with more than 3 vgg blocks are examined since more blocks would help the network extract better features in the image. However, after 4 vgg blocks the accuracy converge and



(a) Baseline model accuracy

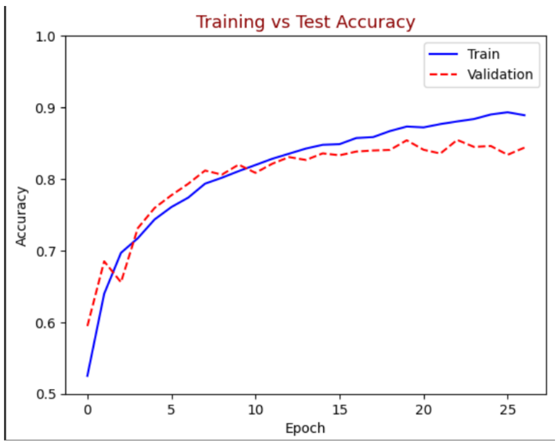


(b) My model accuracy

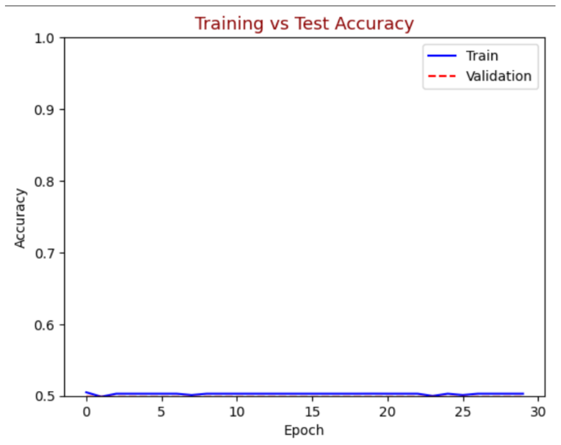
Figure 2: Training and validation set accuracy

stays at around 85% along with a substantial increase in training time. The situation is even worst when I tried to stack 2 convoltional layers in one vgg block The performance drops to around 50% since the network become overly sensitive with the training set. Another structure is to have max pooling layer in only the first 2 or the last two of the vgg blocks in the hidden layer. The former network out perform the latter by 10% in both train and validation set. It because having early max pooling layer filter the noise and selects important features for the later stage in the network, and then those features will be use for a more in-depth inspection in the model.

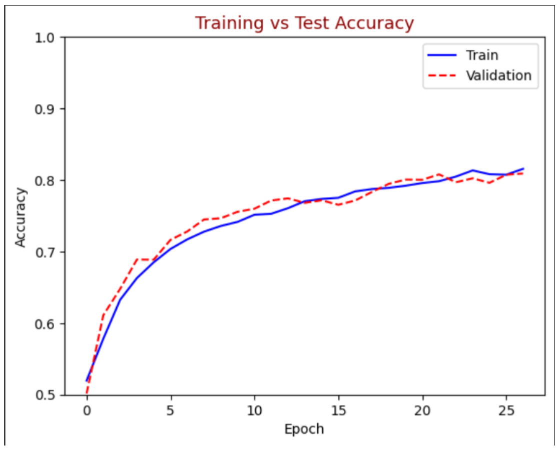
By having drop out in each of the hidden layer, the model is less overfitting indicates with the decrease in the gap between train and validation accuracy. Stable range is from 0.1 to 0.3, drop out rate that is more than 0.4 cause the decrease in performance in validation set. Normal Adam without learning rate decay is outperformed by the model with exponential decay learning rate. Best performance of normal Adam is 83%.



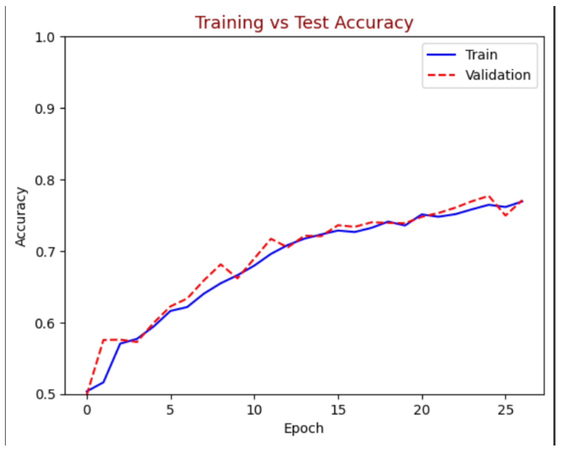
(a) 5 vgg blocks



(b) 2 Convolutional layer in each vgg block



(c) 2 max pooling in the first 2 blocks



(d) 2 max pooling in the last 2 blocks

Figure 3: Training and validation set accuracy of different architect

5.2 K Fold Cross-Validation

The goal of k-fold Cross Validation in CNNs is to examone the performance and the ability for a model to generalize of a neural network on a given dataset. In k-fold cross-validation, the dataset is divided into k folds of equal size, with k-1 of the folds used for model training and the remaining folds for model validation. Each fold is utilized as the validation set once over the course of this operation, which is repeated k times. We can get more precise estimates of the model's performance and lower the chance that the model would be overfit to the training data by employing k-fold cross-validation. By evaluating the model on several, independent subsets of the dataset, we may assess how well it generalizes to new, unexplored data. For this data set, I implement a 5-fold Cross Validation on my benchmark model and my model. For benchmark vgg model, the accuracy is 0.581 and the zero-one loss is 0,419. For my model, the accuracy is 0.537 and the zero-one loss is 0.463. The zero-one loss counts the instances of mis-classified samples and divides its by the total number of samples. Better classification performance can achieve with lower zero-one loss. Those values demonstrate that around 40% of all the samples are wrongly classsify. Although the difference between the 2 models are small but both model have high loss value and low prediction accuracy. This means that the model does not perform well when it comes to new data.

6 Conclusion

In this project, I evaluate a simple vgg model and a modified version of the model for image classification. It was demonstrated that there is a fine line between making a stable model and over-fit or under fit one. I prioritize the efficiency and effectiveness of the model while striking for the balance in bias and variance. Despite the fact that my model performs better than most model's architect I present, it still have problem in generalizing its training to new data and this post as a problem for future project.

References

[1] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[3] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[4] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[5] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[6] Usha Ruby and Vamsidhar Yendapalli. “Binary cross entropy with deep learning technique for image classification”. In: *Int. J. Adv. Trends Comput. Sci. Eng* 9.10 (2020).