Boon Liang Wong, blw219, 01700213

1. CNN Task 1: Classification

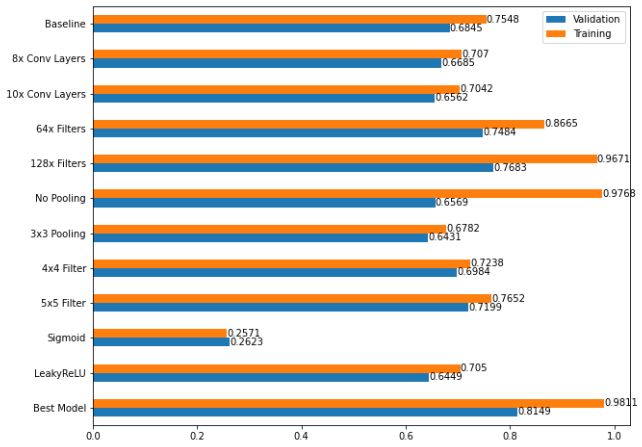

Figure 1. Bar figure with accuracies for different design choices

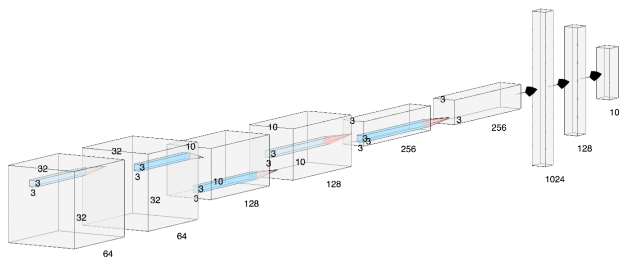

Figure 2. Simplified view of best model; Appendix [A]

With reference to [1], this task was experimented with a baseline model which is a typical shallow network with $Input \rightarrow [Conv \rightarrow ReLU]^{x2} \rightarrow Pool]^{x3} \rightarrow [FC \rightarrow ReLU]^{x2} \rightarrow FC \rightarrow Output$. (68.45% validation acc)

The baseline model consists of 6 convolution layers of 32 depth, 3x3 filters with 2x2 Max Pooling and dense layers of 128,128,10 neurons respectively with softmax activation at the end. Figure 1 illustrates training and validation accuracy for different variation of baseline model.

In figure 1, increasing conv layers in baseline model to 8 and 10 do not increase accuracy. First breakthrough of 6% and 8% in validation accuracy was done when increasing the number of filters in each conv layer to 64 and 128 respectively. When adjusting the filter size to 5x5, slight improvement of 4% was also observed. Interestingly, if no pooling is used, the training accuracy increased to above 95% and validation accuracy drops signifying overfitting. Changing of ReLU activation into Sigmoid also had a detrimental effect of decreasing the baseline accuracy by 42%. The best model architecture found from exp is showed in figure 2 had a validation accuracy of 81.49%; it has 6 conv layers with number of filters increasing with depth (64,128,256 filters num and 3x3 filter size coupling with 2x2 MaxPooling and ReLU) as well as dense layers of 1024,128,10 neurons accordingly with softmax at the end.

2. CNN Task 2: Regression

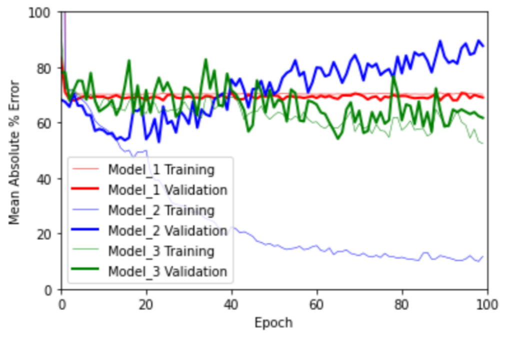

Figure 3. Mean Absolution % Error vs Epochs for different models

Image	Training Error (%)	Validation Error (%)
Kitchen	70.3318	68.8236
Frontal	47.9920	58.4827
Bedroom	70.6273	67.5804
Bathroom	70.2513	69.2935

Table 1. Model_3 results for different images

Model_3 in appendix [B] is the best model with a 58% validation error for frontal image. Model_3 consists of 3 convolution layers of depth 12,32,64 with 5x5 filter size respectively coupled with ReLU activation and MaxPooling with dense layers of 12,5,1 neurons. In table 1, model_3 also performed well on other images. It is likely that model_3 will continue to improve for more Epoch according to fig 3.

In figure 3, the gap between losses for model_2 [C] with 2 conv layers are 80%. Model_2 shows strong overfitting to training data because training loss decreases but validation loss increase. This is also due to small training set, 428 training images. Even though model_1 [D] with 1 conv layer 16 3x3 filter sizes and 2 dense layers has validation error below 75%, however, the results stay flat and did not improve over 100 epochs, implying that no useful feature has been learnt by model_1, hence underfitting.

References

[1] ELEC60009. Lecture 2. *Deep Learning*, page 31, 2021.

Appendix

A. Architecture of Best Model CNN Task 1

```
model = Sequential()
model.add(Conv2D(64, (3, 3),
    padding="same",
      input_shape=x_train.shape[1:]))
model.add(Activation("relu"))
model.add(Conv2D(64, (3, 3),
    padding="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(128, (3, 3),
    padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(128, (3, 3),
    padding="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(256, (3, 3),
    padding="same"))
model.add(Activation("relu"))
model.add(Conv2D(256, (3, 3),
    padding="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(Dense(128))
model.add(Activation("relu"))
model.add(Dense(10))
model.add(Activation("softmax"))
```

B. Architecture of Model_3 CNN Task 2

```
model3.add(Conv2D(12, (5, 5),
   activation="relu",padding="same",
   input_shape=(X_train.shape[1:])))
model3.add(MaxPooling2D((2, 2)))
model3.add(Conv2D(32, (5, 5),
   activation="relu",padding="same"))
model3.add(MaxPooling2D((2, 2)))
model3.add(Conv2D(64, (5, 5),
   activation="relu",padding="same"))
model3.add(MaxPooling2D((2, 2)))

model3.add(Flatten())
model3.add(Dense(12, activation="relu"))
model3.add(Dropout(0.5))
model3.add(Dense(5, activation="relu"))
model3.add(Dense(1, activation="linear"))
```

C. Architecture of Model_2 CNN Task 2

```
model2.add(Conv2D(16, (3, 3),
   activation="relu",padding="same",
   input_shape=(X_train.shape[1:])))
model2.add(MaxPooling2D((2, 2)))
model2.add(Conv2D(32, (3, 3),
   activation="relu",padding="same"))
model2.add(MaxPooling2D((2, 2)))

model2.add(Flatten())
model.add(Dropout(0.5))
model2.add(Dense(10, activation="relu"))
model2.add(Dense(1, activation="linear"))
```

D. Architecture of Model_1 CNN Task 2

```
model.add(Conv2D(16, (3, 3),
   activation="relu",padding="same",
   input_shape=(X_train.shape[1:])))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(5, activation="relu"))
model.add(Dense(1, activation="linear"))
```