CM2: Own Model

Design Choice

In reference to the original default network, the convolutional 2D layer uses the filter size of 32 in the two layers. I wanted to see the effect of changing the number of filters of the convolutional 2D layers and recording the effects on the performance in terms of accuracy on the testing dataset and the cross-entropy loss. Therefore, the structure of the neural network will remain the same and only the filter size will be altered to see the effect on performance. The structure of one of the models is shown in Figure 9.

```
cm2_models[2].summary()

Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 28, 28, 56)        560

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 56)       0
 2D)

 conv2d_7 (Conv2D)           (None, 14, 14, 56)        28280

 flatten_3 (Flatten)         (None, 10976)             0

 dense_6 (Dense)             (None, 128)               1405056

 dense_7 (Dense)             (None, 5)                 645

=================================================================
Total params: 1,434,541
Trainable params: 1,434,541
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: Structure of own model

Although different values of filter will be used for each model, both the convolutional layers will have the same number of filters. For this experiment, 4 values of filters were used for each model, ranging from 40 to 64. The same training, validation and testing datasets were used in each model and the models were trained. For each model, epoch was set to 10 and batch size was 128.

This was achieved using a for loop which is shown Figure 10.

```
cm2_models = [0] * 4

for i in range(4):

  # initialize the default CNN model
  cm2_models[i] = tf.keras.models.Sequential()

  # Add convolution 2D
  cm2_models[i].add(layers.Conv2D(((i*8)+40), kernel_size=(3, 3), activation='relu',
                kernel_initializer='he_normal', padding="same",
                strides=(1,1), input_shape=(28, 28, 1)))

  # Add a pooling layer
  cm2_models[i].add(layers.MaxPooling2D((2, 2)))

  # Add convolution 2D
  cm2_models[i].add(layers.Conv2D(((i*8)+40), kernel_size=(3, 3), padding="same",
                strides=(1,1), activation='relu'))

  # Flatten the layer
  cm2_models[i].add(layers.Flatten())

  # Add the fully connected layer
  cm2_models[i].add(layers.Dense(128, activation='relu'))

  # Add the softmax function for the output layer
  cm2_models[i].add(layers.Dense(5, activation='softmax'))


  cm2_models[i].compile(loss=keras.losses.categorical_crossentropy, optimizer='adam',
              metrics=['accuracy'])
```

Figure 10: For loop that will generate 4 models with different number of filters

The structure of all the four models is shown in Figure 11.

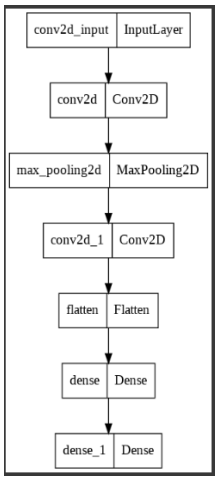

Figure 11:
Visualization of cm2

Due to limitation of computational resources, epoch size was kept small. Otherwise, epoch size should be in the range of 100. The for loop was set up this way so that we can hold multiple models in an array, and it can be easy to manipulate.

The four models are the following:

S. No	Model Number	Number of Filters in Convolution layer
1	0	40
2	1	48
3	2	56
4	3	64

Table 2: Basic layout of the models

All the 4 models will follow the same structure which is shown in Figure 9 and Figure 11, just with different number of filters.

All the models were trained using the same training and validation set which were used for the default network. An example of the code for one of the models is shown in Figure 12:

```
[ ]  cm2_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
       filepath=file_path[0],
       save_weights_only=True,
       verbose=1)
     cm2_model_0 = cm2_models[0].fit(fashion_x_train, fashion_y_train, batch_size=128, epochs=5, verbose=1, validation_data=(fashion_x_val,fashion_y_val),callbacks=[cm2_checkpoint_callb
```

Figure 12: example of training a model

The best model was model number 2 in which there were 56 filters in each convolutional 2D layer. It was selected due to having the best validation and testing accuracy. The comparison in performance among different models is shown in CM3.