In [18]:
```
print("Fashion MNIST train -  rows:",X_train.shape[0]," columns:", X_train.
print("Fashion MNIST valid -  rows:",X_val.shape[0]," columns:", X_val.shap
print("Fashion MNIST test -  rows:",X_test.shape[0]," columns:", X_test.sha
```

```
Fashion MNIST train -  rows: 48000  columns: (28, 28, 1)
Fashion MNIST valid -  rows: 12000  columns: (28, 28, 1)
Fashion MNIST test -  rows: 10000  columns: (28, 28, 1)
```

Type *Markdown* and LaTeX: α^2

Answer [CM1]

Classify the data using a Convolutional Neural Network

###(Default Network)

Below is the model definition as provided in the Assignment Refer to model summary for more information

In [20]:
```
# Model
model = Sequential()
# Add convolution 2D
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 kernel_initializer='he_normal',
                 input_shape=(IMG_ROWS, IMG_COLS, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(32,
                 kernel_size=(3, 3),
                 activation='relu'))
# model.add(MaxPooling2D(pool_size=(2, 2)))
# model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(NUM_CLASSES, activation='softmax'))


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

In [21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 32)        9248
_________________________________________________________________
flatten (Flatten)            (None, 3872)              0
_________________________________________________________________
dense (Dense)                (None, 128)               495744
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 505,957
Trainable params: 505,957
Non-trainable params: 0
_________________________________________________________________
```

In [22]:
```
train_model_default = model.fit(X_train, y_train,
                  batch_size=BATCH_SIZE,
                  epochs=NO_EPOCHS,
                  verbose=1,
                  validation_data=(X_val, y_val))
```

```
accuracy: 0.9892 - val_loss: 0.2337 - val_accuracy: 0.9399
Epoch 15/20
375/375 [==============================] - 5s 14ms/step - loss: 0.0256 -
accuracy: 0.9913 - val_loss: 0.2461 - val_accuracy: 0.9392
Epoch 16/20
375/375 [==============================] - 5s 13ms/step - loss: 0.0248 -
accuracy: 0.9917 - val_loss: 0.2722 - val_accuracy: 0.9339
Epoch 17/20
375/375 [==============================] - 5s 13ms/step - loss: 0.0220 -
accuracy: 0.9925 - val_loss: 0.2799 - val_accuracy: 0.9353
Epoch 18/20
375/375 [==============================] - 5s 13ms/step - loss: 0.0186 -
accuracy: 0.9937 - val_loss: 0.2779 - val_accuracy: 0.9398
Epoch 19/20
375/375 [==============================] - 5s 13ms/step - loss: 0.0140 -
accuracy: 0.9955 - val_loss: 0.3020 - val_accuracy: 0.9405
Epoch 20/20
375/375 [==============================] - 5s 13ms/step - loss: 0.0131 -
accuracy: 0.9952 - val_loss: 0.3053 - val_accuracy: 0.9408
```

Maximum training accuracy is 99.52 % at epoch = 20

Maximum validation accuracy is 94.08% % at epoch = 20 **bold text**

In [23]:
```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.3479614853858948
Test accuracy: 0.9355000257492065
```

Our Test loss is .34 and our Test accuracy is 93.55 %

Here validation training accuracy is near to 100% while validation loss is slightly increasing after few epochs. We will discuss this trend in [CM3]

As we can see in the above graph that after 10 epochs, there is increase in loss for validation set, so there is a chance of model getting overfitted.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: