In [74]:

```
#Fashion mnist

from keras.datasets import fashion_mnist

```

In [75]:

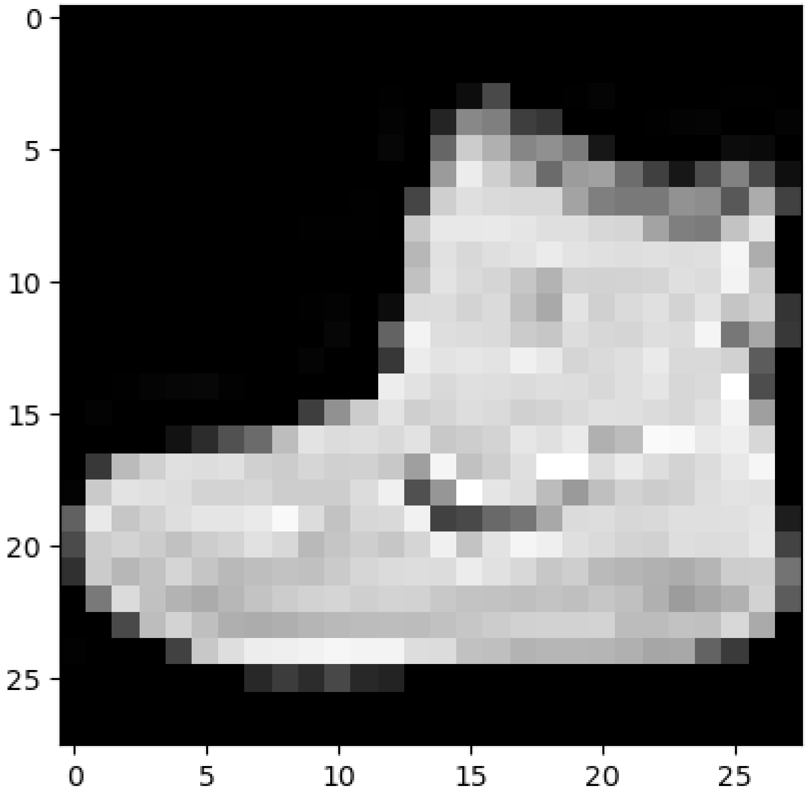
```
(Xtrain, ytrain), (Xtest, ytest) = fashion_mnist.load_data()
```

In [76]:

```
import matplotlib.pyplot as plt

plt.imshow(Xtrain[0], cmap=plt.get_cmap('gray'))
```

Out[76]:

```
<matplotlib.image.AxesImage at 0x2b52ff19490>
```



In [77]:

```
Xtrain.shape
```

Out[77]:

```
(60000, 28, 28)
```

In [78]:

```
ytrain.shape
```

Out[78]:

```
(60000,)
```

In [79]:

```
Xtest.shape
```

Out[79]:

```
(10000, 28, 28)
```

In [80]:

```
ytest.shape
```

Out[80]:

```
(10000,)
```

ANN

In [81]:

```
#Use ANN on this data and make prediction and calculate prediction performance of th
```

In [82]:

```
'''
Find out number of classes-10
Use ANN on this data and make prediction and calculate prediction performance of the
-loss?
-metrics?
-units?
=activation?
-layers?
-optimizer?
-epochs?
'''
```

Out[82]:

```
'\nFind out number of classes-10\nUse ANN on this data and make prediction
and calculate prediction performance of the ANN on test set\n-loss?\n-metr
ics?\n-units?\n=activation?\n-layers?\n-optimizer?\n-epochs?\n'
```

Scaled

In [83]:

```
#0-255
Xtrain_scaled = Xtrain / 255.0
Xtest_scaled = Xtest/255.0
```

In [84]:

```
Xtrain_scaled.shape
```

Out[84]:

```
(60000, 28, 28)
```

In [85]:

```
Xtest_scaled.shape
```

Out[85]:

```
(10000, 28, 28)
```

In [86]:

```
# one hot encoding
from keras.utils import to_categorical

ytrain_ohe = to_categorical(ytrain, num_classes=10)
ytest_ohe = to_categorical(ytest, num_classes=10)
```

In [87]:

```
ytrain_ohe.shape
```

Out[87]:

```
(60000, 10)
```

In [88]:

```
ytest_ohe.shape
```

Out[88]:

```
(10000, 10)
```

ANN

In [89]:

```
from keras.models import Sequential
from keras.layers import Dense, Flatten

fashionANN = Sequential()

```

In [90]:

```
fashionANN
```

Out[90]:

```
<keras.engine.sequential.Sequential at 0x2b52ff13fa0>
```

In [91]:

```
#To convert image 2d to 1d - original [28by28] to [1by784]
fashionANN.add(Flatten())
```

In [92]:

```
#hidden layer
fashionANN.add(Dense(units=256, activation='relu')) #linear
```

In [93]:

```
#hidden layer
fashionANN.add(Dense(units=128, activation='relu')) #linear
```

In [94]:

```
#hidden layer
fashionANN.add(Dense(units=64, activation='relu')) #linear
```

In [95]:

```
#output layer
fashionANN.add(Dense(units=10, activation='softmax'))
```

In [96]:

```
fashionANN.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accu
```

In [97]:

```
fashionANN.fit
```

Out[97]:

```
<bound method Model.fit of <keras.engine.sequential.Sequential object at 0
x000002B52FF13FA0>>
```

In [98]:

```
history = fashionANN.fit(Xtrain_scaled, ytrain_ohe, epochs=5)
```

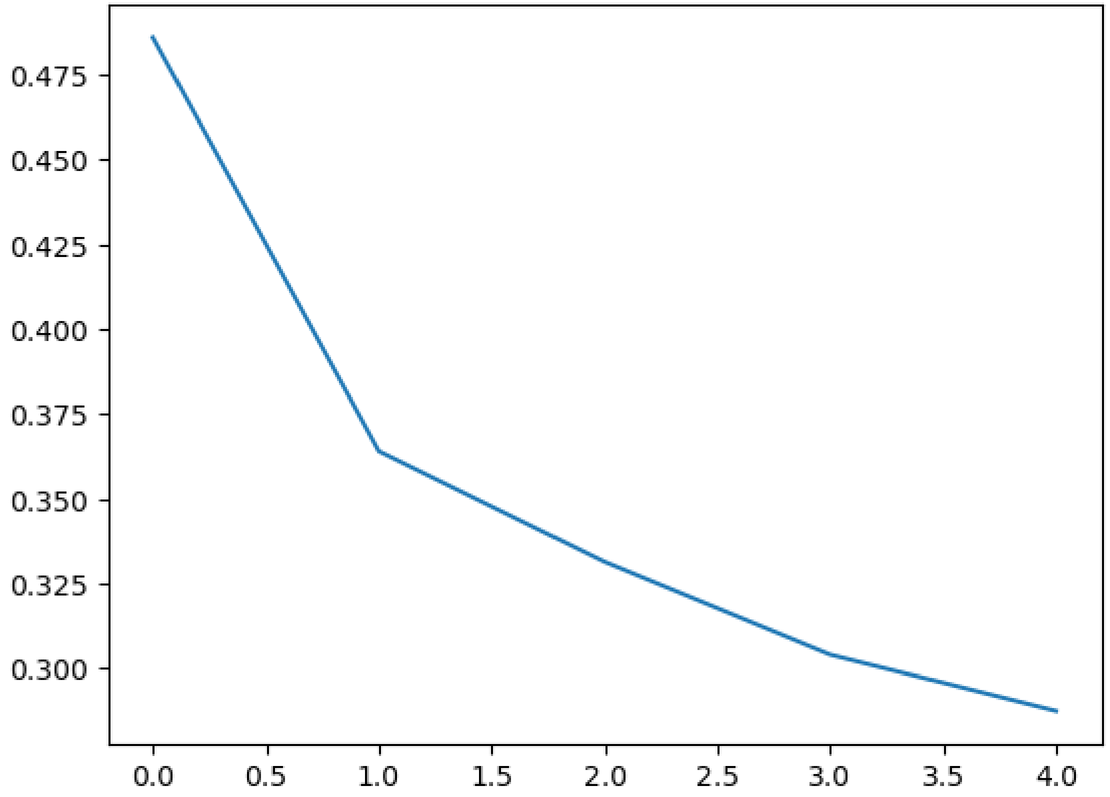
```
Epoch 1/5
1875/1875 [==============================] - 7s 3ms/step - loss: 0.4859 -
accuracy: 0.8250
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3641 -
accuracy: 0.8662
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3314 -
accuracy: 0.8784
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3041 -
accuracy: 0.8882
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2875 -
accuracy: 0.8923
```

In [99]:

```
import matplotlib.pyplot as plt
```

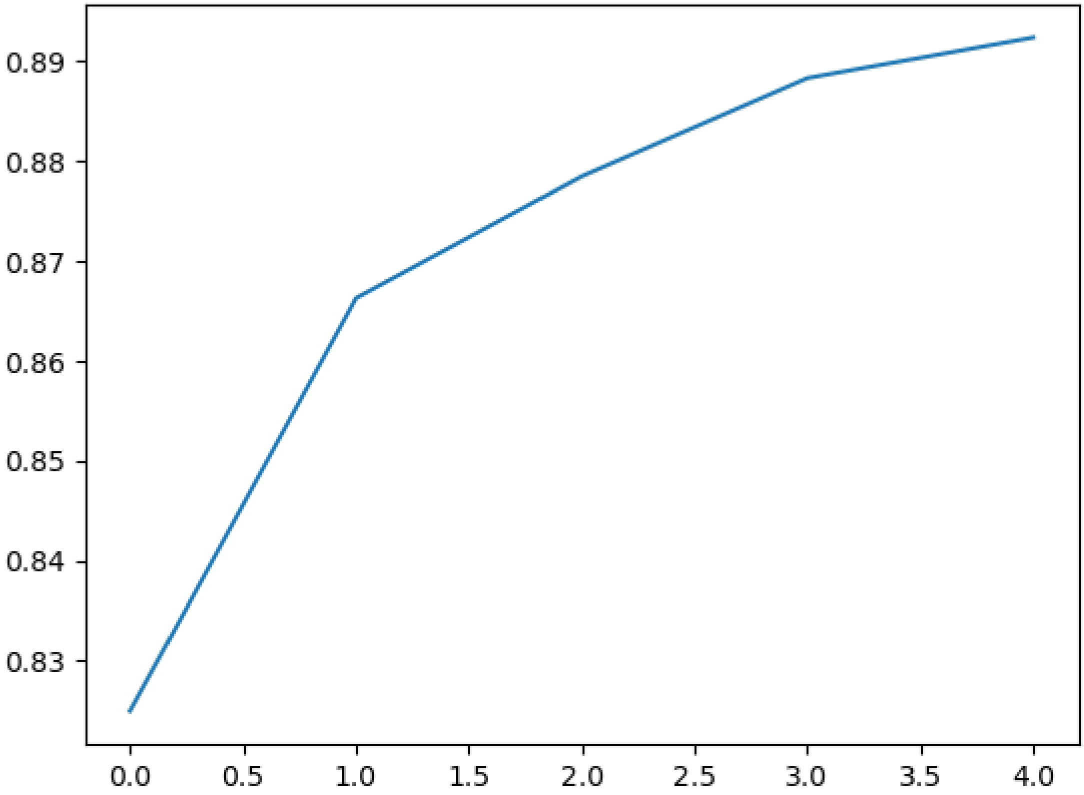
In [100]:

```
plt.plot(history.history['loss'])
plt.show()
```



In [101]:

```
plt.plot(history.history['accuracy'])
plt.show()
```



In [102]:

```
fashionANN.evaluate(Xtest_scaled, ytest_ohe)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3312 - ac
curacy: 0.8806
```

Out[102]:

```
[0.3312040865421295, 0.8805999755859375]
```

ANN2

In [189]:

```
#Fashion mnist

from keras.datasets import fashion_mnist
```

In [190]:

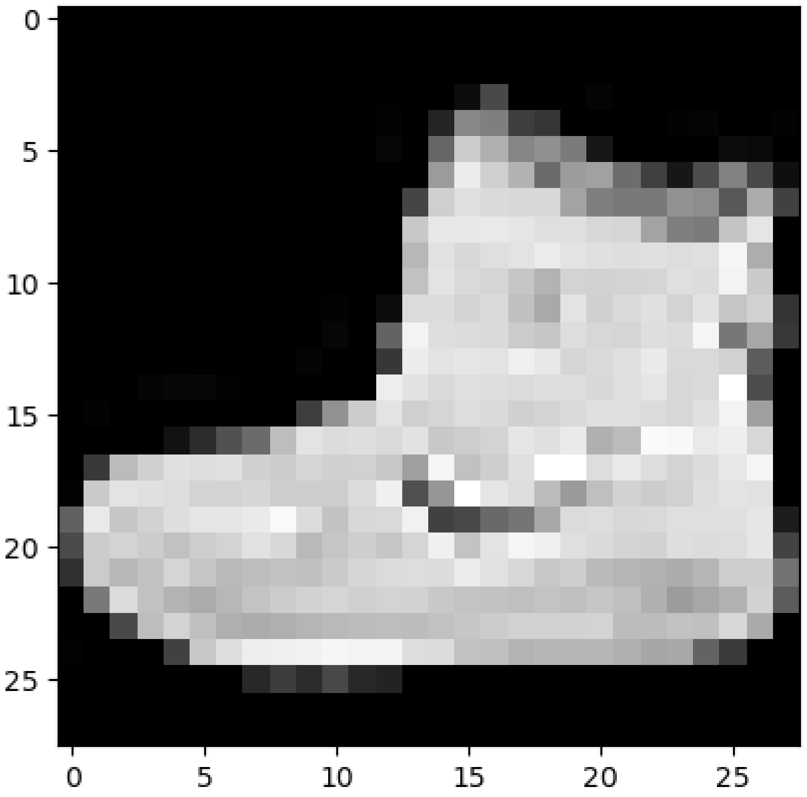
```
(Xtrain, ytrain), (Xtest, ytest) = fashion_mnist.load_data()
```

In [191]:

```
import matplotlib.pyplot as plt

plt.imshow(Xtrain[0], cmap=plt.get_cmap('gray'))
```

Out[191]:

```
<matplotlib.image.AxesImage at 0x2b52ed6be50>
```



In [192]:

```
Xtrain.shape
```

Out[192]:

```
(60000, 28, 28)
```

In [193]:

```
ytest.shape
```

Out[193]:

```
(10000,)
```

In [194]:

```
Xtest.shape
```

Out[194]:

```
(10000, 28, 28)
```

In [195]:

```
ytest.shape
```

Out[195]:

```
(10000,)
```

In [196]:

```
ytrain[0]
```

Out[196]:

```
9
```

In [197]:

```
min(ytest)
```

Out[197]:

```
0
```

In [198]:

```
import collections, numpy
counter = collections.Counter(ytrain)
counter
```

Out[198]:

```
Counter({9: 6000,
         0: 6000,
         3: 6000,
         2: 6000,
         7: 6000,
         5: 6000,
         1: 6000,
         6: 6000,
         4: 6000,
         8: 6000})
```

scaled

In [199]:

```
#0-255
Xtrain_scaled = Xtrain / 255.0
Xtest_scaled = Xtest/255.0
```

In [200]:

```
from keras.utils import to_categorical

ytrain_ohe = to_categorical(ytrain, num_classes=10)
ytest_ohe = to_categorical(ytest, num_classes=10)
```

ANN

In [201]:

```
from keras.models import Sequential
from keras.layers import Dense, Flatten

fashionANN = Sequential()
```

In [202]:

```
#To convert image 2d to 1d - original [28by28] to [1by784]
fashionANN.add(Flatten())
#hidden layer
fashionANN.add(Dense(units=105, activation='relu')) #linear
#output layer
fashionANN.add(Dense(units=10, activation='softmax'))
```

In [203]:

```
fashionANN.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accu
```

In [204]:

```
fashionANN.fit
```

Out[204]:

```
<bound method Model.fit of <keras.engine.sequential.Sequential object at 0
x000002B52ED13940>>
```

In [206]:

```
history = fashionANN.fit(Xtrain_scaled, ytrain_ohe, epochs=5, validation_split=0.2)
```

```
Epoch 1/5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2846 -
accuracy: 0.8954 - val_loss: 0.2919 - val_accuracy: 0.8953
Epoch 2/5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2714 -
accuracy: 0.8983 - val_loss: 0.2860 - val_accuracy: 0.8935
Epoch 3/5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2638 -
accuracy: 0.9019 - val_loss: 0.2893 - val_accuracy: 0.8955
Epoch 4/5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2524 -
accuracy: 0.9048 - val_loss: 0.2808 - val_accuracy: 0.8995
Epoch 5/5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2440 -
accuracy: 0.9094 - val_loss: 0.3098 - val_accuracy: 0.8855
```

In [207]:

```
#ANN architecture

fashionANN.summary()
```

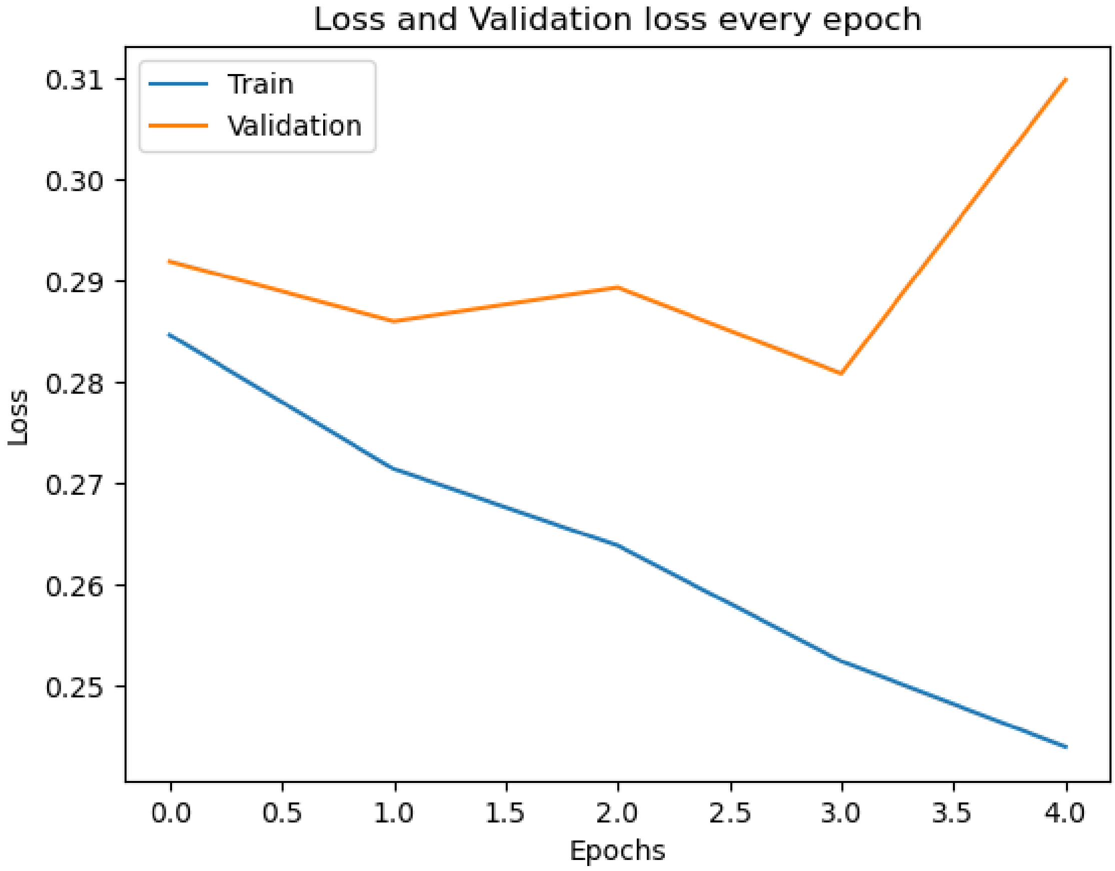
```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_11 (Flatten)        (32, 784)                 0

 dense_41 (Dense)            (32, 105)                 82425

 dense_42 (Dense)            (32, 10)                  1060

=================================================================
Total params: 83,485
Trainable params: 83,485
Non-trainable params: 0
_________________________________________________________________
```

In [210]:

```
import matplotlib.pyplot as plt
```

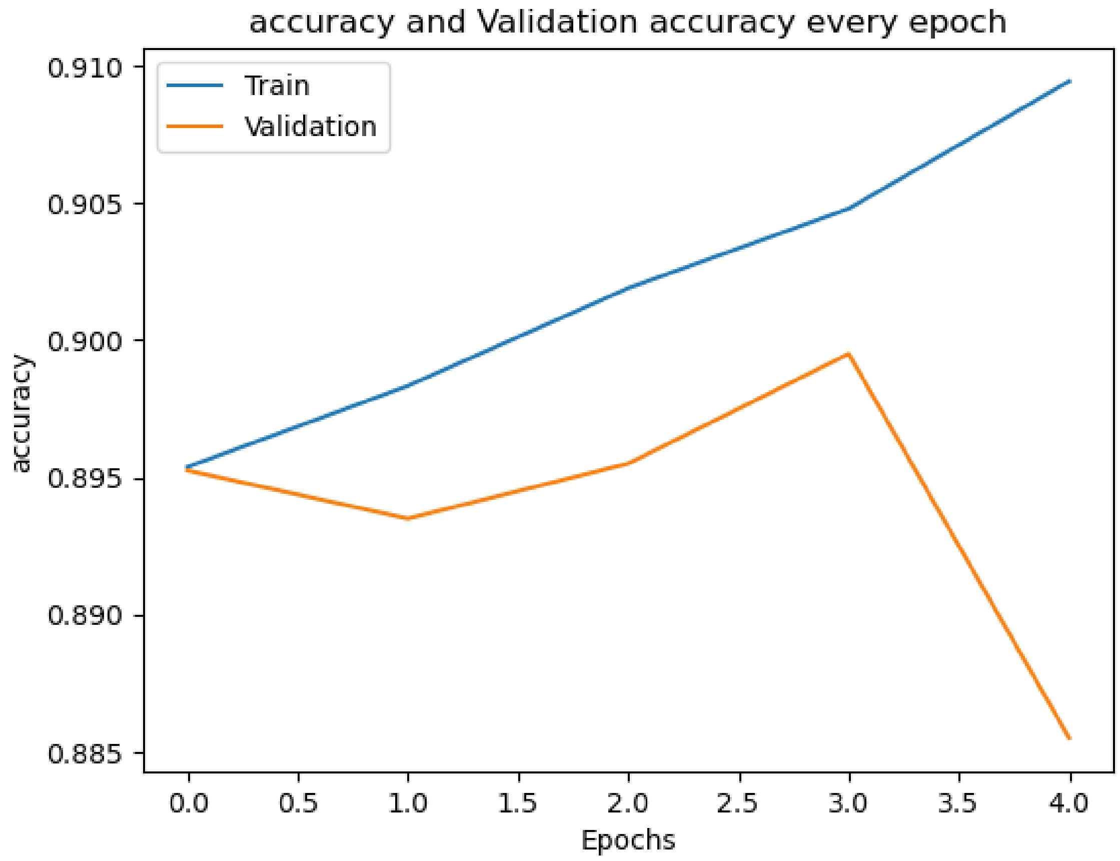
In [211]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'])
plt.title('Loss and Validation loss every epoch')
plt.show()
```



In [212]:

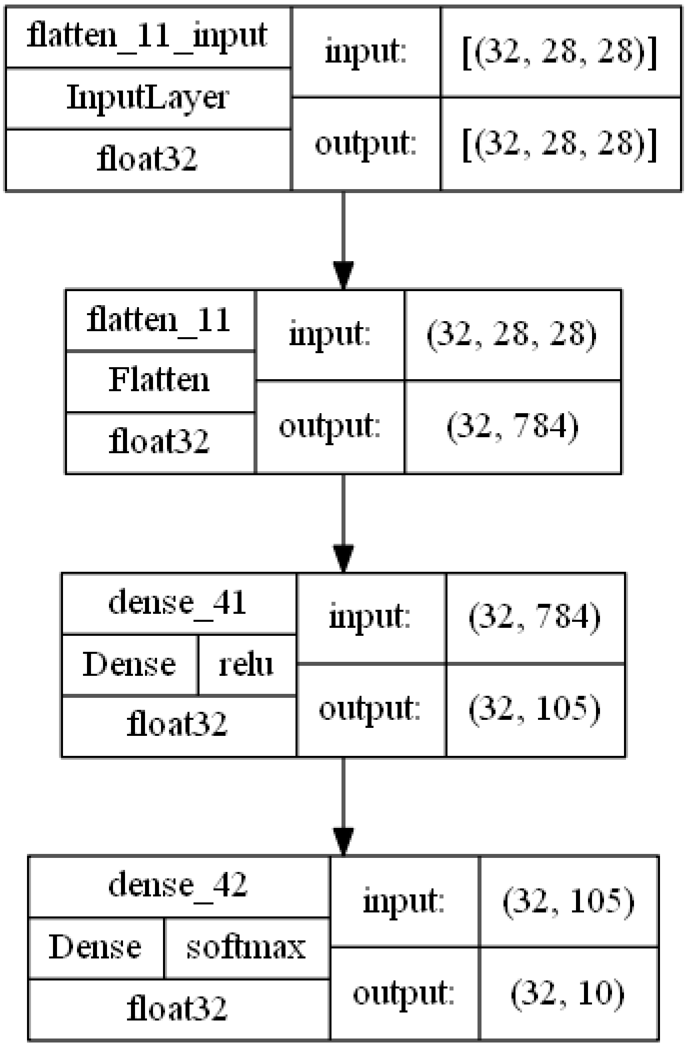
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.legend(['Train', 'Validation'])
plt.title('accuracy and Validation accuracy every epoch')
plt.show()
```



In [213]:

```
from tensorflow.keras.utils import plot_model
plot_model(fashionANN, show_shapes=True, show_dtype=True, show_layer_activations=Tru
```

Out[213]:



In [214]:

```
ypred = fashionANN.predict(Xtest_scaled)
```

```
313/313 [==============================] - 0s 933us/step
```

In [215]:

```
Xtest_scaled[0].shape
```

Out[215]:

```
(28, 28)
```

In [217]:

```
import numpy as np
```

In [218]:

```
sampleimage = np.reshape(Xtest_scaled[0],(1,28,28))
```

In [219]:

```
ypred_first = fashionANN.predict(sampleimage)
```

```
1/1 [==============================] - 0s 20ms/step
```

In [221]:

```
ypred_first
```

Out[221]:

```
array([[6.3418395e-09, 4.1385329e-11, 3.3403996e-08, 8.4393381e-11,
        2.2114696e-07, 4.9443017e-03, 1.1917374e-07, 4.4127041e-03,
        8.4838149e-08, 9.9064249e-01]], dtype=float32)
```

In [222]:

```
ypredclasses_first = np.argmax(ypred_first, axis=-1)
ypredclasses_first
```

Out[222]:

```
array([9], dtype=int64)
```

In [223]:

```
ypredclasses = np.argmax(ypred, axis=-1)
```

In [224]:

```
ypredclasses
```

Out[224]:

```
array([9, 2, 1, ..., 8, 1, 5], dtype=int64)
```

In [225]:

```
ypredclasses.shape
```

Out[225]:

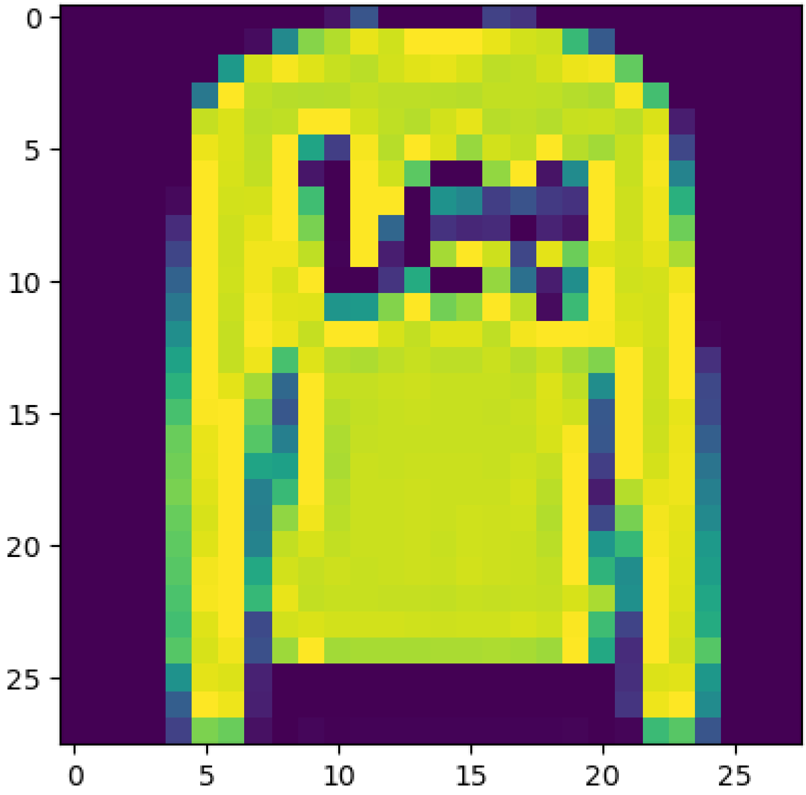
```
(10000,)
```

In [226]:

```
plt.imshow(Xtest[1])
```

Out[226]:

```
<matplotlib.image.AxesImage at 0x2b530698af0>
```



In [227]:

```
ytest[1]
```

Out[227]:

```
2
```

In [228]:

```
ypredclasses[1]
```

Out[228]:

```
2
```

In [230]:

```
from keras.callbacks import ModelCheckpoint

mc = ModelCheckpoint(filepath='bestmodel.h5', monitor = 'val_accuracy', verbose= 1,
```

In [232]:

```
history = fashionANN.fit(Xtrain_scaled, ytrain_ohe, epochs=5, validation_split=0.2,
```

```
Epoch 1/5
1471/1500 [============================>.] - ETA: 0s - loss: 0.2385 - accu
racy: 0.9107
Epoch 1: val_accuracy improved from -inf to 0.89658, saving model to bestm
odel.h5
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2391 -
accuracy: 0.9103 - val_loss: 0.2870 - val_accuracy: 0.8966
Epoch 2/5
1479/1500 [============================>.] - ETA: 0s - loss: 0.2283 - accu
racy: 0.9138
Epoch 2: val_accuracy did not improve from 0.89658
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2281 -
accuracy: 0.9139 - val_loss: 0.2952 - val_accuracy: 0.8947
Epoch 3/5
1481/1500 [============================>.] - ETA: 0s - loss: 0.2221 - accu
racy: 0.9180
Epoch 3: val_accuracy did not improve from 0.89658
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2225 -
accuracy: 0.9179 - val_loss: 0.3040 - val_accuracy: 0.8922
Epoch 4/5
1491/1500 [============================>.] - ETA: 0s - loss: 0.2160 - accu
racy: 0.9179
Epoch 4: val_accuracy did not improve from 0.89658
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2165 -
accuracy: 0.9176 - val_loss: 0.3017 - val_accuracy: 0.8936
Epoch 5/5
1492/1500 [============================>.] - ETA: 0s - loss: 0.2091 - accu
racy: 0.9224
Epoch 5: val_accuracy did not improve from 0.89658
1500/1500 [==============================] - 3s 2ms/step - loss: 0.2093 -
accuracy: 0.9223 - val_loss: 0.3075 - val_accuracy: 0.8909
```

In [233]:

```
fashionANN.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_11 (Flatten)        (None, 784)               0

 dense_41 (Dense)            (None, 105)               82425

 dense_42 (Dense)            (None, 10)                1060

=================================================================
Total params: 83,485
Trainable params: 83,485
Non-trainable params: 0
_________________________________________________________________
```

In [236]:

```
##from keras.models import load_model
##bestmodel = load_model('/content/bestmodel.h5')            #having error
```

In [237]:

```
fashionANN.evaluate(Xtest_scaled, ytest_ohe)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.3551 - ac
curacy: 0.8801
```

Out[237]:

```
[0.355119913816452, 0.8801000118255615]
```

In [239]:

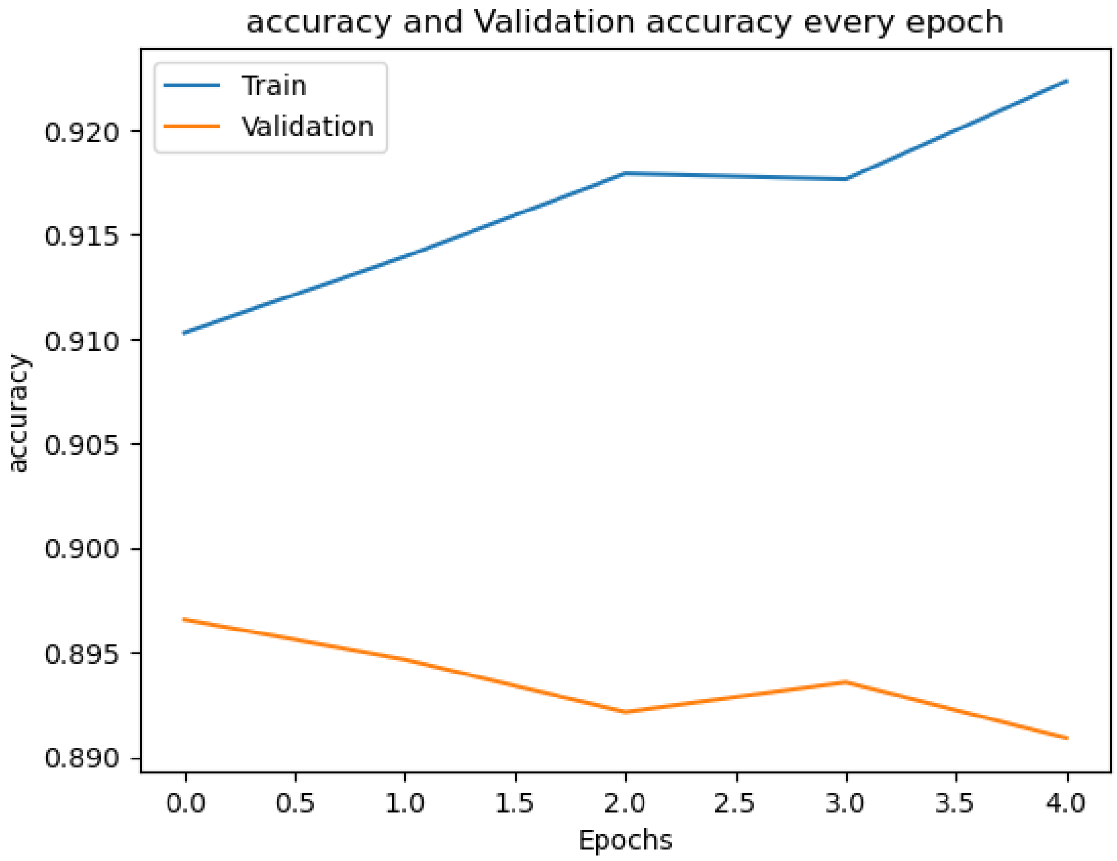
```
##bestmodel.evaluate(Xtest_scaled, ytest_ohe)        #so having error
```

In [240]:

```
import matplotlib.pyplot as plt
```

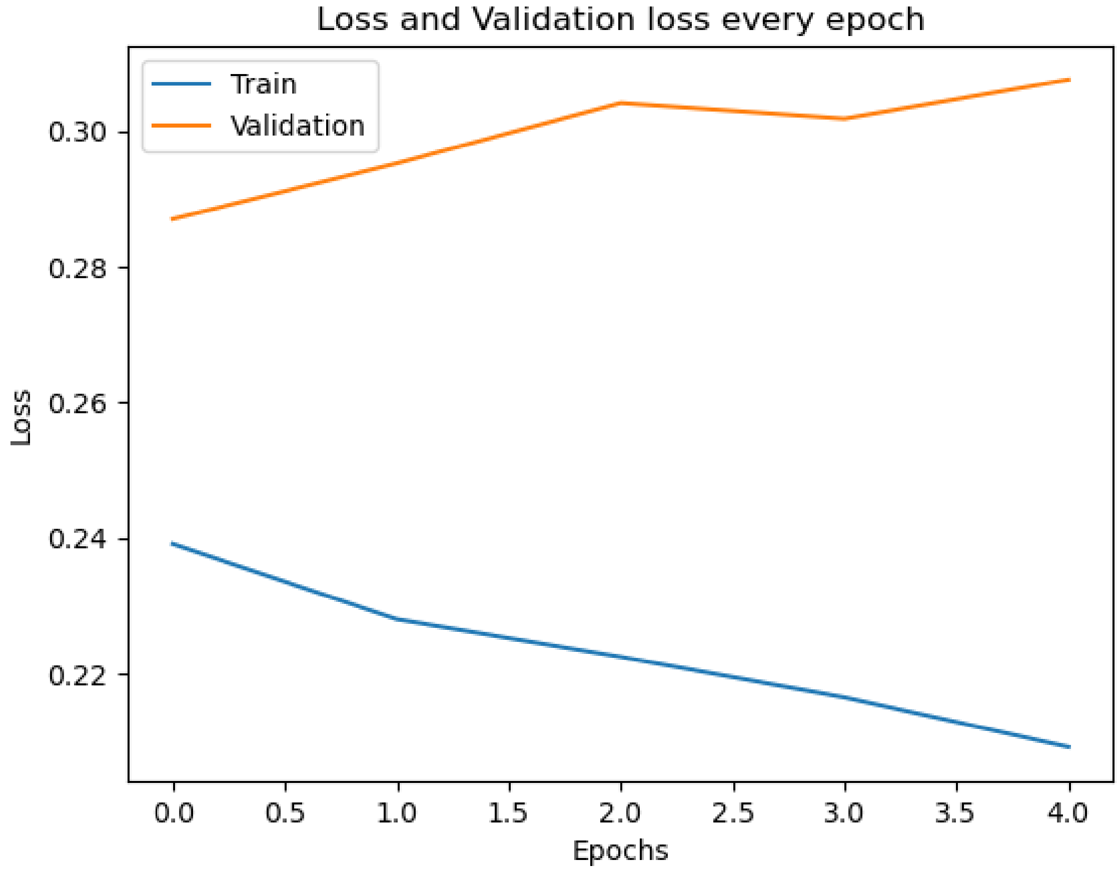
In [241]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.legend(['Train', 'Validation'])
plt.title('accuracy and Validation accuracy every epoch')
plt.show()
```



In [242]:

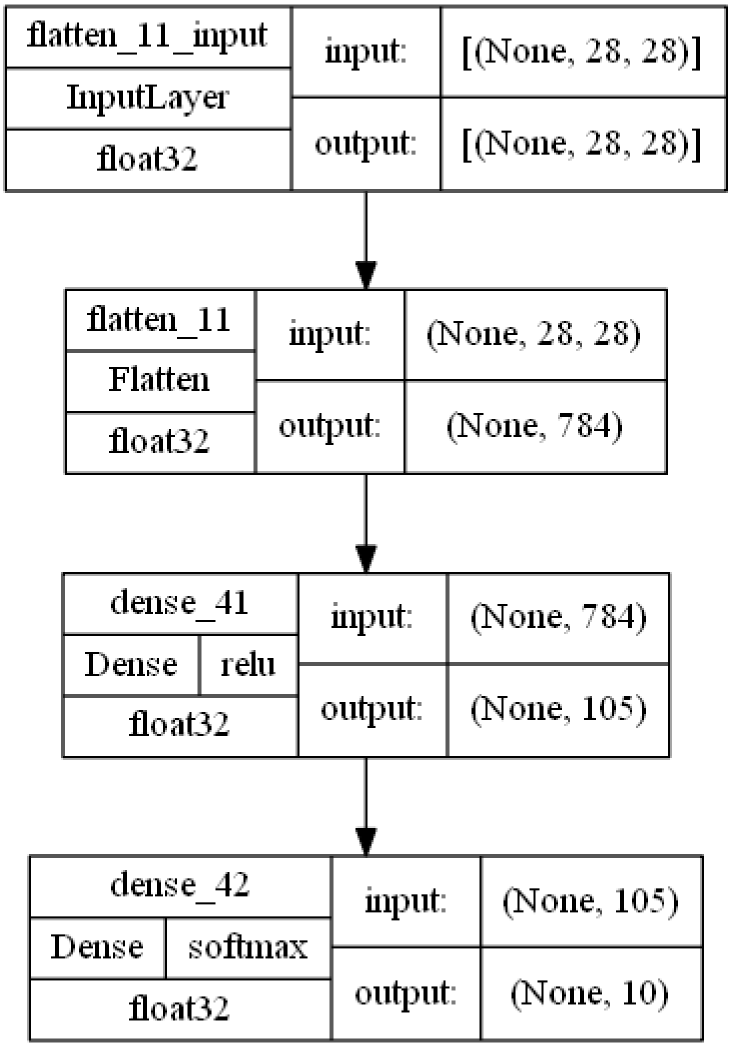
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'])
plt.title('Loss and Validation loss every epoch')
plt.show()
```



In [243]:

```python
from tensorflow.keras.utils import plot_model
plot_model(fashionANN, show_shapes=True, show_dtype=True, show_layer_activations=Tru
```

Out[243]:



In [244]:

```python
fashionANN.evaluate(Xtest_scaled, ytest_ohe)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.3551 - ac
curacy: 0.8801
```

Out[244]:

```
[0.355119913816452, 0.8801000118255615]
```

In [ ]:

```python

```