In [1]:

```python
import tensorflow as tf
#import tensorflow_datasets as tfds
```

Baseline model

In [2]:

```python
# Load Data
# tfds.list_builders()
# (train, test), info = tfds.load("mnist",split=['train', 'test'], with_info=True, as_supervised=True)
#(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
40960/29515 [==========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
16384/5148 [===============================================================================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

In [3]:

```python
# Baseline model definition
model = tf.keras.Sequential([
                             tf.keras.layers.Flatten(input_shape=(28, 28)),
                             tf.keras.layers.Dense(16, activation='relu'),
                             tf.keras.layers.Dense(16, activation='relu'),
                             tf.keras.layers.Dense(10, activation='softmax')
])
```

In [4]:

```python
# Baseline model compilation
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy']
             )
```

In [5]:

```python
# Baseline model fitting
history = model.fit(x_train, y_train,
                    batch_size=128,
                    epochs=100,
                    validation_data=(x_test, y_test),
                    verbose=2
                    )
```

```
Epoch 1/100
469/469 - 2s - loss: 3.2091 - sparse_categorical_accuracy: 0.2202 - val_loss: 1.9478 - val_sparse_categorical_accuracy: 0.2667 - 2s/epoch - 4ms/step
Epoch 2/100
469/469 - 1s - loss: 1.6610 - sparse_categorical_accuracy: 0.3797 - val_loss: 1.3970 - val_sparse_categorical_accuracy: 0.4779 - 1s/epoch - 2ms/step
Epoch 3/100
469/469 - 1s - loss: 1.2795 - sparse_categorical_accuracy: 0.5152 - val_loss: 1.1449 - val_sparse_categorical_accuracy: 0.5865 - 1s/epoch - 2ms/step
Epoch 4/100
469/469 - 1s - loss: 0.9778 - sparse_categorical_accuracy: 0.6406 - val_loss: 0.9231 - val_sparse_categorical_accuracy: 0.6622 - 1s/epoch - 2ms/step
Epoch 5/100
469/469 - 1s - loss: 0.8393 - sparse_categorical_accuracy: 0.6777 - val_loss: 0.8620 - val_sparse_categorical_accuracy: 0.6791 - 1s/epoch - 2ms/step
Epoch 6/100
469/469 - 1s - loss: 0.7918 - sparse_categorical_accuracy: 0.6902 - val_loss: 0.8111 - val_sparse_categorical_accuracy: 0.6731 - 1s/epoch - 2ms/step
Epoch 7/100
469/469 - 1s - loss: 0.7735 - sparse_categorical_accuracy: 0.6992 - val_loss: 0.8310 - val_sparse_categorical_accuracy: 0.6915 - 1s/epoch - 2ms/step
Epoch 8/100
469/469 - 1s - loss: 0.7459 - sparse_categorical_accuracy: 0.7121 - val_loss: 0.7439 - val_sparse_categorical_accuracy: 0.7107 - 1s/epoch - 2ms/step
Epoch 9/100
469/469 - 1s - loss: 0.7016 - sparse_categorical_accuracy: 0.7241 - val_loss: 0.7359 - val_sparse_categorical_accuracy: 0.7129 - 1s/epoch - 2ms/step
Epoch 10/100
469/469 - 1s - loss: 0.6915 - sparse_categorical_accuracy: 0.7268 - val_loss: 0.7291 - val_sparse_categorical_accuracy: 0.7096 - 1s/epoch - 2ms/step
Epoch 11/100
469/469 - 1s - loss: 0.6724 - sparse_categorical_accuracy: 0.7290 - val_loss: 0.6907 - val_sparse_categorical_accuracy: 0.7246 - 1s/epoch - 2ms/step
Epoch 12/100
469/469 - 1s - loss: 0.6649 - sparse_categorical_accuracy: 0.7315 - val_loss: 0.6859 - val_sparse_categorical_accuracy: 0.7246 - 1s/epoch - 2ms/step
Epoch 13/100
469/469 - 1s - loss: 0.6541 - sparse_categorical_accuracy: 0.7327 - val_loss: 0.6741 - val_sparse_categorical_accuracy: 0.7290 - 1s/epoch - 2ms/step
Epoch 14/100
469/469 - 1s - loss: 0.6445 - sparse_categorical_accuracy: 0.7352 - val_loss: 0.7007 - val_sparse_categorical_accuracy: 0.7242 - 1s/epoch - 2ms/step
Epoch 15/100
469/469 - 1s - loss: 0.6434 - sparse_categorical_accuracy: 0.7364 - val_loss: 0.6666 - val_sparse_categorical_accuracy: 0.7248 - 1s/epoch - 2ms/step
Epoch 16/100
469/469 - 1s - loss: 0.6352 - sparse_categorical_accuracy: 0.7389 - val_loss: 0.7348 - val_sparse_categorical_accuracy: 0.7268 - 1s/epoch - 2ms/step
Epoch 17/100
469/469 - 1s - loss: 0.6303 - sparse_categorical_accuracy: 0.7426 - val_loss: 0.6772 - val_sparse_categorical_accuracy: 0.7259 - 1s/epoch - 2ms/step
Epoch 18/100
469/469 - 1s - loss: 0.6201 - sparse_categorical_accuracy: 0.7448 - val_loss: 0.7010 - val_sparse_categorical_accuracy: 0.7146 - 1s/epoch - 2ms/step
Epoch 19/100
469/469 - 1s - loss: 0.6296 - sparse_categorical_accuracy: 0.7406 - val_loss: 0.6678 - val_sparse_categorical_accuracy: 0.7311 - 1s/epoch - 2ms/step
Epoch 20/100
469/469 - 1s - loss: 0.6157 - sparse_categorical_accuracy: 0.7445 - val_loss: 0.6691 - val_sparse_categorical_accuracy: 0.7320 - 1s/epoch - 2ms/step
Epoch 21/100
469/469 - 1s - loss: 0.6096 - sparse_categorical_accuracy: 0.7445 - val_loss: 0.6650 - val_sparse_categorical_accuracy: 0.7319 - 1s/epoch - 2ms/step
Epoch 22/100
469/469 - 1s - loss: 0.6183 - sparse_categorical_accuracy: 0.7420 - val_loss: 0.6726 - val_sparse_categorical_accuracy: 0.7228 - 1s/epoch - 2ms/step
Epoch 23/100
469/469 - 1s - loss: 0.6153 - sparse_categorical_accuracy: 0.7448 - val_loss: 0.6668 - val_sparse_categorical_accuracy: 0.7328 - 1s/epoch - 2ms/step
Epoch 24/100
469/469 - 1s - loss: 0.6180 - sparse_categorical_accuracy: 0.7438 - val_loss: 0.6703 - val_sparse_categorical_accuracy: 0.7287 - 1s/epoch - 2ms/step
Epoch 25/100
469/469 - 1s - loss: 0.6045 - sparse_categorical_accuracy: 0.7462 - val_loss: 0.6516 - val_sparse_categorical_accuracy: 0.7342 - 1s/epoch - 2ms/step
Epoch 26/100
469/469 - 1s - loss: 0.6080 - sparse_categorical_accuracy: 0.7463 - val_loss: 0.6848 - val_sparse_categorical_accuracy: 0.7270 - 1s/epoch - 2ms/step
Epoch 27/100
469/469 - 1s - loss: 0.6084 - sparse_categorical_accuracy: 0.7449 - val_loss: 0.6732 - val_sparse_categorical_accuracy: 0.7352 - 1s/epoch - 2ms/step
Epoch 28/100
469/469 - 1s - loss: 0.6017 - sparse_categorical_accuracy: 0.7480 - val_loss: 0.7623 - val_sparse_categorical_accuracy: 0.6961 - 1s/epoch - 2ms/step
Epoch 29/100
469/469 - 1s - loss: 0.6027 - sparse_categorical_accuracy: 0.7443 - val_loss: 0.6593 - val_sparse_categorical_accuracy: 0.7280 - 1s/epoch - 2ms/step
Epoch 30/100
469/469 - 1s - loss: 0.6004 - sparse_categorical_accuracy: 0.7466 - val_loss: 0.6697 - val_sparse_categorical_accuracy: 0.7354 - 1s/epoch - 2ms/step
Epoch 31/100
469/469 - 1s - loss: 0.5920 - sparse_categorical_accuracy: 0.7507 - val_loss: 0.6432 - val_sparse_categorical_accuracy: 0.7380 - 1s/epoch - 2ms/step
Epoch 32/100
469/469 - 1s - loss: 0.6010 - sparse_categorical_accuracy: 0.7476 - val_loss: 0.6642 - val_sparse_categorical_accuracy: 0.7300 - 1s/epoch - 2ms/step
Epoch 33/100
469/469 - 1s - loss: 0.5984 - sparse_categorical_accuracy: 0.7480 - val_loss: 0.6796 - val_sparse_categorical_accuracy: 0.7218 - 1s/epoch - 2ms/step
Epoch 34/100
469/469 - 1s - loss: 0.5998 - sparse_categorical_accuracy: 0.7482 - val_loss: 0.6926 - val_sparse_categorical_accuracy: 0.6881 - 1s/epoch - 2ms/step
Epoch 35/100
469/469 - 1s - loss: 0.6006 - sparse_categorical_accuracy: 0.7487 - val_loss: 0.6377 - val_sparse_categorical_accuracy: 0.7371 - 1s/epoch - 2ms/step
Epoch 36/100
469/469 - 1s - loss: 0.5894 - sparse_categorical_accuracy: 0.7513 - val_loss: 0.6563 - val_sparse_categorical_accuracy: 0.7254 - 1s/epoch - 3ms/step
Epoch 37/100
469/469 - 1s - loss: 0.5987 - sparse_categorical_accuracy: 0.7476 - val_loss: 0.6667 - val_sparse_categorical_accuracy: 0.7343 - 1s/epoch - 2ms/step
Epoch 38/100
469/469 - 1s - loss: 0.5942 - sparse_categorical_accuracy: 0.7495 - val_loss: 0.6445 - val_sparse_categorical_accuracy: 0.7305 - 1s/epoch - 2ms/step
Epoch 39/100
469/469 - 1s - loss: 0.5911 - sparse_categorical_accuracy: 0.7500 - val_loss: 0.6673 - val_sparse_categorical_accuracy: 0.7356 - 1s/epoch - 2ms/step
Epoch 40/100
469/469 - 1s - loss: 0.6023 - sparse_categorical_accuracy: 0.7481 - val_loss: 0.6358 - val_sparse_categorical_accuracy: 0.7399 - 1s/epoch - 2ms/step
Epoch 41/100
469/469 - 1s - loss: 0.5911 - sparse_categorical_accuracy: 0.7506 - val_loss: 0.6474 - val_sparse_categorical_accuracy: 0.7362 - 1s/epoch - 2ms/step
Epoch 42/100
469/469 - 1s - loss: 0.5830 - sparse_categorical_accuracy: 0.7533 - val_loss: 0.6369 - val_sparse_categorical_accuracy: 0.7389 - 1s/epoch - 2ms/step
Epoch 43/100
469/469 - 1s - loss: 0.5953 - sparse_categorical_accuracy: 0.7494 - val_loss: 0.6637 - val_sparse_categorical_accuracy: 0.7290 - 1s/epoch - 2ms/step
Epoch 44/100
469/469 - 1s - loss: 0.5918 - sparse_categorical_accuracy: 0.7497 - val_loss: 0.6398 - val_sparse_categorical_accuracy: 0.7400 - 1s/epoch - 2ms/step
Epoch 45/100
469/469 - 1s - loss: 0.5840 - sparse_categorical_accuracy: 0.7509 - val_loss: 0.6614 - val_sparse_categorical_accuracy: 0.7403 - 1s/epoch - 2ms/step
Epoch 46/100
469/469 - 1s - loss: 0.5837 - sparse_categorical_accuracy: 0.7537 - val_loss: 0.6401 - val_sparse_categorical_accuracy: 0.7380 - 1s/epoch - 2ms/step
Epoch 47/100
469/469 - 1s - loss: 0.5847 - sparse_categorical_accuracy: 0.7536 - val_loss: 0.7288 - val_sparse_categorical_accuracy: 0.7106 - 1s/epoch - 3ms/step
Epoch 48/100
469/469 - 2s - loss: 0.5846 - sparse_categorical_accuracy: 0.7523 - val_loss: 0.6442 - val_sparse_categorical_accuracy: 0.7369 - 2s/epoch - 3ms/step
Epoch 49/100
469/469 - 1s - loss: 0.5842 - sparse_categorical_accuracy: 0.7515 - val_loss: 0.6514 - val_sparse_categorical_accuracy: 0.7317 - 1s/epoch - 2ms/step
Epoch 50/100
469/469 - 1s - loss: 0.5804 - sparse_categorical_accuracy: 0.7520 - val_loss: 0.6840 - val_sparse_categorical_accuracy: 0.7130 - 1s/epoch - 2ms/step
Epoch 51/100
469/469 - 1s - loss: 0.5782 - sparse_categorical_accuracy: 0.7526 - val_loss: 0.6617 - val_sparse_categorical_accuracy: 0.7330 - 1s/epoch - 2ms/step
Epoch 52/100
469/469 - 1s - loss: 0.5962 - sparse_categorical_accuracy: 0.7478 - val_loss: 0.6421 - val_sparse_categorical_accuracy: 0.7414 - 1s/epoch - 2ms/step
Epoch 53/100
469/469 - 1s - loss: 0.5942 - sparse_categorical_accuracy: 0.7488 - val_loss: 0.6575 - val_sparse_categorical_accuracy: 0.7287 - 1s/epoch - 2ms/step
Epoch 54/100
469/469 - 1s - loss: 0.5764 - sparse_categorical_accuracy: 0.7517 - val_loss: 0.6297 - val_sparse_categorical_accuracy: 0.7380 - 1s/epoch - 2ms/step
Epoch 55/100
469/469 - 1s - loss: 0.5804 - sparse_categorical_accuracy: 0.7521 - val_loss: 0.6369 - val_sparse_categorical_accuracy: 0.7399 - 1s/epoch - 2ms/step
Epoch 56/100
469/469 - 1s - loss: 0.5763 - sparse_categorical_accuracy: 0.7549 - val_loss: 0.6368 - val_sparse_categorical_accuracy: 0.7411 - 1s/epoch - 2ms/step
Epoch 57/100
469/469 - 1s - loss: 0.5717 - sparse_categorical_accuracy: 0.7538 - val_loss: 0.6301 - val_sparse_categorical_accuracy: 0.7374 - 1s/epoch - 2ms/step
Epoch 58/100
469/469 - 1s - loss: 0.5676 - sparse_categorical_accuracy: 0.7574 - val_loss: 0.6593 - val_sparse_categorical_accuracy: 0.7421 - 1s/epoch - 2ms/step
Epoch 59/100
469/469 - 1s - loss: 0.5809 - sparse_categorical_accuracy: 0.7524 - val_loss: 0.6282 - val_sparse_categorical_accuracy: 0.7438 - 1s/epoch - 2ms/step
Epoch 60/100
469/469 - 1s - loss: 0.5729 - sparse_categorical_accuracy: 0.7549 - val_loss: 0.6556 - val_sparse_categorical_accuracy: 0.7303 - 1s/epoch - 2ms/step
Epoch 61/100
469/469 - 1s - loss: 0.5843 - sparse_categorical_accuracy: 0.7503 - val_loss: 0.6394 - val_sparse_categorical_accuracy: 0.7366 - 1s/epoch - 2ms/step
Epoch 62/100
469/469 - 1s - loss: 0.5801 - sparse_categorical_accuracy: 0.7514 - val_loss: 0.6298 - val_sparse_categorical_accuracy: 0.7344 - 1s/epoch - 2ms/step
Epoch 63/100
469/469 - 1s - loss: 0.5803 - sparse_categorical_accuracy: 0.7533 - val_loss: 0.7300 - val_sparse_categorical_accuracy: 0.6976 - 1s/epoch - 2ms/step
Epoch 64/100
469/469 - 1s - loss: 0.5759 - sparse_categorical_accuracy: 0.7551 - val_loss: 0.6296 - val_sparse_categorical_accuracy: 0.7417 - 1s/epoch - 2ms/step
Epoch 65/100
469/469 - 1s - loss: 0.5738 - sparse_categorical_accuracy: 0.7555 - val_loss: 0.6516 - val_sparse_categorical_accuracy: 0.7343 - 1s/epoch - 2ms/step
Epoch 66/100
469/469 - 1s - loss: 0.5813 - sparse_categorical_accuracy: 0.7534 - val_loss: 0.6474 - val_sparse_categorical_accuracy: 0.7329 - 1s/epoch - 2ms/step
Epoch 67/100
469/469 - 1s - loss: 0.5659 - sparse_categorical_accuracy: 0.7576 - val_loss: 0.6311 - val_sparse_categorical_accuracy: 0.7388 - 1s/epoch - 2ms/step
Epoch 68/100
469/469 - 1s - loss: 0.5642 - sparse_categorical_accuracy: 0.7563 - val_loss: 0.6257 - val_sparse_categorical_accuracy: 0.7442 - 1s/epoch - 2ms/step
Epoch 69/100
469/469 - 1s - loss: 0.5752 - sparse_categorical_accuracy: 0.7559 - val_loss: 0.6203 - val_sparse_categorical_accuracy: 0.7436 - 1s/epoch - 2ms/step
Epoch 70/100
469/469 - 1s - loss: 0.5673 - sparse_categorical_accuracy: 0.7567 - val_loss: 0.6349 - val_sparse_categorical_accuracy: 0.7444 - 1s/epoch - 2ms/step
Epoch 71/100
469/469 - 1s - loss: 0.5698 - sparse_categorical_accuracy: 0.7562 - val_loss: 0.6226 - val_sparse_categorical_accuracy: 0.7445 - 1s/epoch - 2ms/step
Epoch 72/100
469/469 - 1s - loss: 0.5721 - sparse_categorical_accuracy: 0.7559 - val_loss: 0.6504 - val_sparse_categorical_accuracy: 0.7421 - 1s/epoch - 2ms/step
Epoch 73/100
469/469 - 1s - loss: 0.5709 - sparse_categorical_accuracy: 0.7574 - val_loss: 0.6141 - val_sparse_categorical_accuracy: 0.7465 - 1s/epoch - 2ms/step
Epoch 74/100
469/469 - 1s - loss: 0.5680 - sparse_categorical_accuracy: 0.7578 - val_loss: 0.6338 - val_sparse_categorical_accuracy: 0.7402 - 1s/epoch - 2ms/step
Epoch 75/100
469/469 - 1s - loss: 0.5616 - sparse_categorical_accuracy: 0.7580 - val_loss: 0.6487 - val_sparse_categorical_accuracy: 0.7347 - 1s/epoch - 2ms/step
Epoch 76/100
469/469 - 1s - loss: 0.5643 - sparse_categorical_accuracy: 0.7585 - val_loss: 0.6472 - val_sparse_categorical_accuracy: 0.7408 - 1s/epoch - 2ms/step
Epoch 77/100
469/469 - 1s - loss: 0.5605 - sparse_categorical_accuracy: 0.7593 - val_loss: 0.7401 - val_sparse_categorical_accuracy: 0.7312 - 1s/epoch - 2ms/step
Epoch 78/100
469/469 - 1s - loss: 0.5745 - sparse_categorical_accuracy: 0.7566 - val_loss: 0.6475 - val_sparse_categorical_accuracy: 0.7391 - 1s/epoch - 2ms/step
Epoch 79/100
469/469 - 1s - loss: 0.5636 - sparse_categorical_accuracy: 0.7600 - val_loss: 0.6492 - val_sparse_categorical_accuracy: 0.7412 - 1s/epoch - 2ms/step
Epoch 80/100
469/469 - 1s - loss: 0.5715 - sparse_categorical_accuracy: 0.7578 - val_loss: 0.6314 - val_sparse_categorical_accuracy: 0.7449 - 1s/epoch - 2ms/step
Epoch 81/100
469/469 - 1s - loss: 0.5682 - sparse_categorical_accuracy: 0.7591 - val_loss: 0.6291 - val_sparse_categorical_accuracy: 0.7477 - 1s/epoch - 2ms/step
Epoch 82/100
469/469 - 1s - loss: 0.5638 - sparse_categorical_accuracy: 0.7594 - val_loss: 0.6203 - val_sparse_categorical_accuracy: 0.7448 - 1s/epoch - 2ms/step
Epoch 83/100
469/469 - 1s - loss: 0.5608 - sparse_categorical_accuracy: 0.7613 - val_loss: 0.6217 - val_sparse_categorical_accuracy: 0.7457 - 1s/epoch - 2ms/step
Epoch 84/100
469/469 - 1s - loss: 0.5589 - sparse_categorical_accuracy: 0.7621 - val_loss: 0.6211 - val_sparse_categorical_accuracy: 0.7505 - 1s/epoch - 2ms/step
Epoch 85/100
469/469 - 1s - loss: 0.5576 - sparse_categorical_accuracy: 0.7631 - val_loss: 0.6178 - val_sparse_categorical_accuracy: 0.7471 - 1s/epoch - 2ms/step
Epoch 86/100
469/469 - 1s - loss: 0.5638 - sparse_categorical_accuracy: 0.7617 - val_loss: 0.6497 - val_sparse_categorical_accuracy: 0.7450 - 1s/epoch - 2ms/step
Epoch 87/100
469/469 - 1s - loss: 0.5713 - sparse_categorical_accuracy: 0.7596 - val_loss: 0.6227 - val_sparse_categorical_accuracy: 0.7473 - 1s/epoch - 2ms/step
Epoch 88/100
469/469 - 1s - loss: 0.5635 - sparse_categorical_accuracy: 0.7644 - val_loss: 0.6350 - val_sparse_categorical_accuracy: 0.7476 - 1s/epoch - 2ms/step
Epoch 89/100
469/469 - 1s - loss: 0.5604 - sparse_categorical_accuracy: 0.7634 - val_loss: 0.6221 - val_sparse_categorical_accuracy: 0.7495 - 994ms/epoch - 2ms/step
Epoch 90/100
469/469 - 1s - loss: 0.5553 - sparse_categorical_accuracy: 0.7665 - val_loss: 0.6377 - val_sparse_categorical_accuracy: 0.7467 - 1s/epoch - 2ms/step
Epoch 91/100
469/469 - 1s - loss: 0.5632 - sparse_categorical_accuracy: 0.7638 - val_loss: 0.6066 - val_sparse_categorical_accuracy: 0.7516 - 1s/epoch - 2ms/step
Epoch 92/100
469/469 - 1s - loss: 0.5548 - sparse_categorical_accuracy: 0.7646 - val_loss: 0.6194 - val_sparse_categorical_accuracy: 0.7477 - 1s/epoch - 2ms/step
Epoch 93/100
469/469 - 1s - loss: 0.5482 - sparse_categorical_accuracy: 0.7667 - val_loss: 0.6469 - val_sparse_categorical_accuracy: 0.7450 - 1s/epoch - 2ms/step
Epoch 94/100
469/469 - 1s - loss: 0.5734 - sparse_categorical_accuracy: 0.7619 - val_loss: 0.6237 - val_sparse_categorical_accuracy: 0.7500 - 1s/epoch - 2ms/step
Epoch 95/100
469/469 - 1s - loss: 0.5598 - sparse_categorical_accuracy: 0.7671 - val_loss: 0.6189 - val_sparse_categorical_accuracy: 0.7495 - 1s/epoch - 2ms/step
Epoch 96/100
469/469 - 1s - loss: 0.5660 - sparse_categorical_accuracy: 0.7618 - val_loss: 0.6517 - val_sparse_categorical_accuracy: 0.7442 - 1s/epoch - 2ms/step
Epoch 97/100
469/469 - 1s - loss: 0.5727 - sparse_categorical_accuracy: 0.7597 - val_loss: 0.6193 - val_sparse_categorical_accuracy: 0.7477 - 1s/epoch - 2ms/step
Epoch 98/100
469/469 - 1s - loss: 0.5669 - sparse_categorical_accuracy: 0.7624 - val_loss: 0.6263 - val_sparse_categorical_accuracy: 0.7523 - 1s/epoch - 2ms/step
Epoch 99/100
469/469 - 1s - loss: 0.5514 - sparse_categorical_accuracy: 0.7660 - val_loss: 0.6449 - val_sparse_categorical_accuracy: 0.7534 - 1s/epoch - 2ms/step
Epoch 100/100
469/469 - 1s - loss: 0.5657 - sparse_categorical_accuracy: 0.7639 - val_loss: 0.6534 - val_sparse_categorical_accuracy: 0.7434 - 1s/epoch - 3ms/step
```

In [6]:

```python
# Baseline model evaluation
model.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 0s - loss: 0.6534 - sparse_categorical_accuracy: 0.7434 - 361ms/epoch - 1ms/step
```

Out[6]: `[0.6533612012863159, 0.743399977684021]`

The baseline model with `kernel_initializer='glorot_uniform'`, `bias_initializer='zeros'`

Try different models

Model1: `adam` optimizer with learning rate=e^{-3}, `random_uniform` initializer, `dropout` regularization with rate=0.1.

In [7]:

```python
model1 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.1),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_uniform')
])
```

In [8]:

```python
model1.compile(optimizer=tf.keras.optimizers.Adam(0.001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history1 = model1.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [10]:

```python
model1.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4092 - sparse_categorical_accuracy: 0.8486 - 621ms/epoch - 2ms/step
```

Out[10]: `[0.4091724753379822, 0.8485999703407288]`

Model2: `adam` optimizer with learning rate=e^{-4}, `random_uniform` initializer, `dropout` regularization with rate=0.1.

In [11]:

```python
model2 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.1),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_uniform')
])
```

In [12]:

```python
model2.compile(optimizer=tf.keras.optimizers.Adam(0.0001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history2 = model2.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [14]:

```python
model2.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4488 - sparse_categorical_accuracy: 0.8352 - 637ms/epoch - 2ms/step
```

Out[14]: `[0.44878360629081726, 0.8352000117301941]`

Model3: `adam` optimizer with learning rate=e^{-3}, `random_normal` initializer, `dropout` regularization with rate=0.1.

In [15]:

```python
model3 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.1),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_normal')
])
```

In [16]:

```python
model3.compile(optimizer=tf.keras.optimizers.Adam(0.001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history3 = model3.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [18]:

```python
model3.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4114 - sparse_categorical_accuracy: 0.8471 - 556ms/epoch - 2ms/step
```

Out[18]: `[0.4114220142364502, 0.847100019454956]`

Model4: `adam` optimizer with learning rate=e^{-4}, `random_normal` initializer, `dropout` regularization with rate=0.1.

In [19]:

```python
model4 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.1),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_normal')
])
```

In [20]:

```python
model4.compile(optimizer=tf.keras.optimizers.Adam(0.0001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history4 = model4.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [22]:

```python
model4.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4543 - sparse_categorical_accuracy: 0.8316 - 534ms/epoch - 2ms/step
```

Out[22]: `[0.4542616605758667, 0.83160001039505]`

Model5: `adam` optimizer with learning rate=e^{-3}, `random_uniform` initializer, `dropout` regularization with rate=0.2.

In [23]:

```python
model5 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_uniform')
])
```

In [24]:

```python
model5.compile(optimizer=tf.keras.optimizers.Adam(0.001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history5 = model5.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [26]:

```python
model5.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4243 - sparse_categorical_accuracy: 0.8442 - 577ms/epoch - 2ms/step
```

Out[26]: `[0.4243125021457672, 0.8442000150680542]`

Model6: `adam` optimizer with learning rate=e^{-4}, `random_uniform` initializer, `dropout` regularization with rate=0.2.

In [27]:

```python
model6 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_uniform'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_uniform')
])
```

In [28]:

```python
model6.compile(optimizer=tf.keras.optimizers.Adam(0.0001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history6 = model6.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [30]:

```python
model6.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4614 - sparse_categorical_accuracy: 0.8335 - 521ms/epoch - 2ms/step
```

Out[30]: `[0.4613606035709381, 0.8335000276565552]`

Model7: `adam` optimizer with learning rate=e^{-3}, `random_normal` initializer, `dropout` regularization with rate=0.2.

In [31]:

```python
model7 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_normal')
])
```

In [32]:

```python
model7.compile(optimizer=tf.keras.optimizers.Adam(0.001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history7 = model7.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

In [34]:

```python
model7.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4226 - sparse_categorical_accuracy: 0.8413 - 526ms/epoch - 2ms/step
```

Out[34]: `[0.42263558506965637, 0.8413000106811523]`

Model8: `adam` optimizer with learning rate=e^{-4}, `random_normal` initializer, `dropout` regularization with rate=0.2.

In [35]:

```python
model8 = tf.keras.models.Sequential([
  tf.keras.layers.Input(shape=(28, 28,1)),
  tf.keras.layers.MaxPooling2D((2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(16, activation='relu', kernel_initializer='random_normal'),
  tf.keras.layers.Dense(10, activation='softmax', kernel_initializer='random_normal')
])
```

In [36]:

```python
model8.compile(optimizer=tf.keras.optimizers.Adam(0.0001),
               loss='sparse_categorical_crossentropy',
               metrics=['sparse_categorical_accuracy']
               )
```

In [ ]:

```python
history8 = model8.fit(x_train, y_train,
                      batch_size=128,
                      epochs=100,
                      validation_data=(x_test, y_test),
                      verbose=2
                      )
```

[38]:
```
model8.evaluate(x_test, y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.4646 - sparse_categorical_accuracy: 0.8298 - 529m
s/epoch - 2ms/step
```

Out[38]:
```
[0.46464452147483826, 0.829800009727478]
```

Plots

[39]:
```
import matplotlib.pyplot as plt
```

Plots for different kernel initializers

[40]:
```
training_accuracy0 = history.history['sparse_categorical_accuracy']
validation_accuracy0 = history.history['val_sparse_categorical_accuracy'
]

training_accuracy1 = history1.history['sparse_categorical_accuracy']
validation_accuracy1 = history1.history['val_sparse_categorical_accurac
y']

training_accuracy2 = history2.history['sparse_categorical_accuracy']
validation_accuracy2 = history2.history['val_sparse_categorical_accurac
y']

training_accuracy3 = history3.history['sparse_categorical_accuracy']
validation_accuracy3 = history3.history['val_sparse_categorical_accurac
y']

training_accuracy4 = history4.history['sparse_categorical_accuracy']
validation_accuracy4 = history4.history['val_sparse_categorical_accurac
y']

training_accuracy5 = history5.history['sparse_categorical_accuracy']
validation_accuracy5 = history5.history['val_sparse_categorical_accurac
y']

training_accuracy6 = history6.history['sparse_categorical_accuracy']
validation_accuracy6 = history6.history['val_sparse_categorical_accurac
y']

training_accuracy7 = history7.history['sparse_categorical_accuracy']
validation_accuracy7 = history7.history['val_sparse_categorical_accurac
y']

training_accuracy8 = history8.history['sparse_categorical_accuracy']
validation_accuracy8 = history8.history['val_sparse_categorical_accurac
y']

epochs_range=range(100)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, training_accuracy0, label='Train Acc for Baselin
e')
plt.plot(epochs_range, training_accuracy1, label='Train Acc for Model1')
plt.plot(epochs_range, training_accuracy2, label='Train Acc for Model2')
plt.plot(epochs_range, training_accuracy3, label='Train Acc for Model3')
plt.plot(epochs_range, training_accuracy4, label='Train Acc for Model4')
plt.plot(epochs_range, training_accuracy5, label='Train Acc for Model5')
plt.plot(epochs_range, training_accuracy6, label='Train Acc for Model6')
plt.plot(epochs_range, training_accuracy7, label='Train Acc for Model7')
plt.plot(epochs_range, training_accuracy8, label='Train Acc for Model8')
plt.legend(loc='lower right')
plt.title('Training Accuracy For All Models')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, validation_accuracy0, label='Val Acc for Baselin
e')
plt.plot(epochs_range, validation_accuracy1, label='Val Acc for Model1')
plt.plot(epochs_range, validation_accuracy2, label='Val Acc for Model2')
plt.plot(epochs_range, validation_accuracy3, label='Val Acc for Model3')
plt.plot(epochs_range, validation_accuracy4, label='Val Acc for Model4')
plt.plot(epochs_range, validation_accuracy5, label='Val Acc for Model5')
plt.plot(epochs_range, validation_accuracy6, label='Val Acc for Model6')
plt.plot(epochs_range, validation_accuracy7, label='Val Acc for Model7')
plt.plot(epochs_range, validation_accuracy8, label='Val Acc for Model8')
plt.legend(loc='lower right')
plt.title('Validation Accuracy For All Models')
plt.show()
```