In this model in comparison with mnist-simple-1 two more dense layer with 64 nodes are add to the model and it cause more acuracy and less loss in teraining and testing data.

```
import numpy as np
from tensorflow import keras
import tensorflow as tf
from tensorflow.keras.constraints import max_norm
from tensorflow.keras.utils import to_categorical
from keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Activation, Conv2D, MaxPool2D, Dropout, Flatten
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D, Flatten
from tensorflow.keras.layers import BatchNormalization
from keras.src.engine.training import optimizer
from keras.src.layers.attention.multi_head_attention import activation
import pandas as pd
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
```

```
X_train shape (60000, 28, 28)
y_train shape (60000,)
X_test shape (10000, 28, 28)
y_test shape (10000,)
```

```
# normalize each value for each pixel for the entire vector for each input # Normalize the inputs from 0-255 to between 0 and 1 by dividing by 255
X_test = X_test/255
X_train = X_train/255
```

```
y_train[0]
```

```
5
```

```
y_onehot_train = tf.one_hot(y_train, 10)
```

```
y_onehot_train[0]
```

```
<tf.Tensor: shape=(10,), dtype=float32, numpy=array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)>
```

```
model = tf.keras.models.Sequential([
        layers.Input(X_train.shape[1:]),
        layers.Flatten(),
        layers.Dense(64, activation='elu'),
        layers.Dense(64, activation='elu'),
        layers.Dense(10, activation='softmax')
    ])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

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

 dense (Dense)               (None, 64)                50240

 dense_1 (Dense)             (None, 64)                4160
```

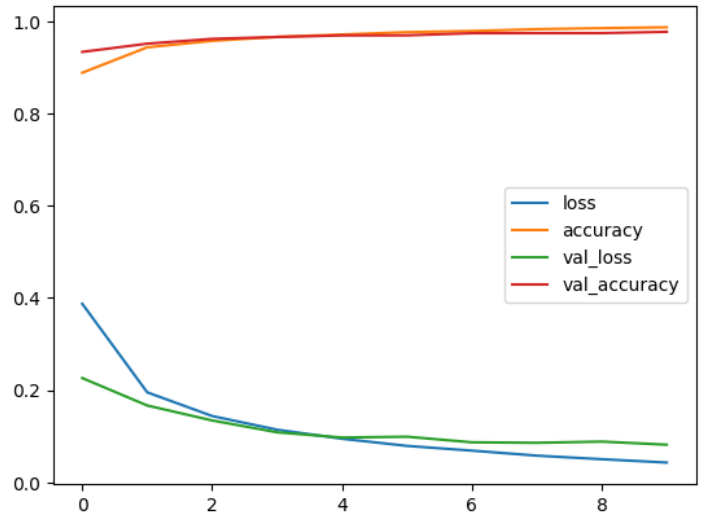
```
 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 55050 (215.04 KB)
Trainable params: 55050 (215.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
history= model.fit(X_train, y_train, epochs=10, batch_size=128, validation_data=(X_test, y_test))
```

```
Epoch 1/10
469/469 [==============================] - 2s 3ms/step - loss: 0.3871 - accuracy: 0.8882 - val_loss: 0.2260 - val_accuracy: 0.9334
Epoch 2/10
469/469 [==============================] - 1s 3ms/step - loss: 0.1951 - accuracy: 0.9434 - val_loss: 0.1665 - val_accuracy: 0.9513
Epoch 3/10
469/469 [==============================] - 1s 3ms/step - loss: 0.1437 - accuracy: 0.9574 - val_loss: 0.1341 - val_accuracy: 0.9617
Epoch 4/10
469/469 [==============================] - 1s 3ms/step - loss: 0.1140 - accuracy: 0.9660 - val_loss: 0.1083 - val_accuracy: 0.9657
Epoch 5/10
469/469 [==============================] - 1s 3ms/step - loss: 0.0944 - accuracy: 0.9711 - val_loss: 0.0970 - val_accuracy: 0.9692
Epoch 6/10
469/469 [==============================] - 1s 3ms/step - loss: 0.0789 - accuracy: 0.9763 - val_loss: 0.0988 - val_accuracy: 0.9693
Epoch 7/10
469/469 [==============================] - 1s 3ms/step - loss: 0.0686 - accuracy: 0.9788 - val_loss: 0.0867 - val_accuracy: 0.9740
Epoch 8/10
469/469 [==============================] - 1s 3ms/step - loss: 0.0578 - accuracy: 0.9827 - val_loss: 0.0856 - val_accuracy: 0.9741
Epoch 9/10
469/469 [==============================] - 1s 3ms/step - loss: 0.0500 - accuracy: 0.9850 - val_loss: 0.0882 - val_accuracy: 0.9741
Epoch 10/10
469/469 [==============================] - 1s 3ms/step - loss: 0.0428 - accuracy: 0.9868 - val_loss: 0.0815 - val_accuracy: 0.9770
```

```
pd.DataFrame(history.history).plot()
```

```
<Axes: >
```



✓ 0s completed at 4:06 PM