```
import pandas as pd
import numpy as np

np.random.seed(1212)

import keras
from keras.models import Model
from keras.layers import *
from keras import optimizers
```

```
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
```

```
df_train.head()
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	p
0	1	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0	
1	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0	
2	1	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0	
3	4	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0	
4	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0	

5 rows × 785 columns

```
df_features = df_train.iloc[:, 1:785]
df_label = df_train.iloc[:, 0]

X_test = df_test.iloc[:, 0:784]

print(X_test.shape)
print(X_train.shape)
print(X_cv.shape)
```

```
(12638, 784)
(7336, 784)
(1834, 784)
```

```
from sklearn.model_selection import train_test_split
X_train, X_cv, y_train, y_cv = train_test_split(df_features, df_label,
                                                test_size = 0.2,
                                                random_state = 1212)

X_train = X_train.to_numpy().reshape(7336, 784)
X_cv = X_cv.to_numpy().reshape(1834, 784)

X_test = X_test.to_numpy().reshape(12638, 784)
```

```
print((min(X_train[1]), max(X_train[1])))
```

```
(0.0, 254.0)
```

```
X_train = X_train.astype('float32'); X_cv= X_cv.astype('float32'); X_test = X_test.astype('float32')
X_train /= 255; X_cv /= 255; X_test /= 255

# Convert labels to One Hot Encoded
num_digits = 10
y_train = keras.utils.to_categorical(y_train, num_digits)
y_cv = keras.utils.to_categorical(y_cv, num_digits)
```

```
print(y_train[0])
print(y_train[3])
```

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```

```
n_input = 784
n_hidden_1 = 300
n_hidden_2 = 100
n_hidden_3 = 100
n_hidden_4 = 200
num_digits = 10
```

```
Inp = Input(shape=(784,))
x = Dense(n_hidden_1, activation='relu', name = "Hidden_Layer_1")(Inp)
x = Dense(n_hidden_2, activation='relu', name = "Hidden_Layer_2")(x)
x = Dense(n_hidden_3, activation='relu', name = "Hidden_Layer_3")(x)
x = Dense(n_hidden_4, activation='relu', name = "Hidden_Layer_4")(x)
output = Dense(num_digits, activation='softmax', name = "Output_Layer")(x)
```

```
model = Model(Inp, output)
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 784)]             0

 Hidden_Layer_1 (Dense)      (None, 300)               235500

 Hidden_Layer_2 (Dense)      (None, 100)               30100

 Hidden_Layer_3 (Dense)      (None, 100)               10100

 Hidden_Layer_4 (Dense)      (None, 200)               20200

 Output_Layer (Dense)        (None, 10)                2010

=================================================================
Total params: 297,910
Trainable params: 297,910
Non-trainable params: 0
```

```
learning_rate = 0.1
training_epochs = 20
batch_size = 100
sgd = optimizers.SGD(lr=learning_rate)
```

```
/usr/local/lib/python3.10/dist-packages/keras/optimizers/legacy/gradient_descent.py:114: UserWarning: The `lr` argument is dep
  super().__init__(name, **kwargs)
```

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

```
history1 = model.fit(X_train, y_train,
                     batch_size = batch_size,
                     epochs = training_epochs,
                     verbose = 2,
                     validation_data=(X_cv, y_cv))
```

```
Epoch 1/20
74/74 - 1s - loss: nan - accuracy: 0.1010 - val_loss: nan - val_accuracy: 0.0911 - 1s/epoch - 20ms/step
Epoch 2/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 489ms/epoch - 7ms/step
Epoch 3/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 463ms/epoch - 6ms/step
Epoch 4/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 461ms/epoch - 6ms/step
Epoch 5/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 507ms/epoch - 7ms/step
Epoch 6/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 564ms/epoch - 8ms/step
Epoch 7/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 489ms/epoch - 7ms/step
Epoch 8/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 498ms/epoch - 7ms/step
Epoch 9/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 506ms/epoch - 7ms/step
Epoch 10/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 477ms/epoch - 6ms/step
Epoch 11/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 591ms/epoch - 8ms/step
Epoch 12/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 453ms/epoch - 6ms/step
Epoch 13/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 491ms/epoch - 7ms/step
Epoch 14/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 601ms/epoch - 8ms/step
Epoch 15/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 803ms/epoch - 11ms/step
Epoch 16/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 836ms/epoch - 11ms/step
Epoch 17/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 755ms/epoch - 10ms/step
Epoch 18/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 499ms/epoch - 7ms/step
Epoch 19/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 474ms/epoch - 6ms/step
Epoch 20/20
74/74 - 0s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 471ms/epoch - 6ms/step
```

```
Inp = Input(shape=(784,))
x = Dense(n_hidden_1, activation='relu', name = "Hidden_Layer_1")(Inp)
x = Dense(n_hidden_2, activation='relu', name = "Hidden_Layer_2")(x)
x = Dense(n_hidden_3, activation='relu', name = "Hidden_Layer_3")(x)
x = Dense(n_hidden_4, activation='relu', name = "Hidden_Layer_4")(x)
output = Dense(num_digits, activation='softmax', name = "Output_Layer")(x)


adam = keras.optimizers.Adam(lr=learning_rate)
model2 = Model(Inp, output)

model2.compile(loss='categorical_crossentropy',
               optimizer='adam',
               metrics=['accuracy'])
```

```
/usr/local/lib/python3.10/dist-packages/keras/optimizers/legacy/adam.py:117: UserWarning: The `lr` argument is deprecated, use
  super().__init__(name, **kwargs)
```

```
history2 = model2.fit(X_train, y_train,
                      batch_size = batch_size,
                      epochs = training_epochs,
                      verbose = 2,
                      validation_data=(X_cv, y_cv))
```

```
Epoch 1/20
74/74 - 2s - loss: nan - accuracy: 0.6366 - val_loss: nan - val_accuracy: 0.0911 - 2s/epoch - 24ms/step
Epoch 2/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 560ms/epoch - 8ms/step
Epoch 3/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 593ms/epoch - 8ms/step
Epoch 4/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 591ms/epoch - 8ms/step
Epoch 5/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 577ms/epoch - 8ms/step
Epoch 6/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 864ms/epoch - 12ms/step
Epoch 7/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 975ms/epoch - 13ms/step
Epoch 8/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 1s/epoch - 14ms/step
Epoch 9/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 597ms/epoch - 8ms/step
Epoch 10/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 573ms/epoch - 8ms/step
Epoch 11/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 587ms/epoch - 8ms/step
Epoch 12/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 567ms/epoch - 8ms/step
Epoch 13/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 558ms/epoch - 8ms/step
Epoch 14/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 529ms/epoch - 7ms/step
Epoch 15/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 574ms/epoch - 8ms/step
Epoch 16/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 542ms/epoch - 7ms/step
Epoch 17/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 575ms/epoch - 8ms/step
Epoch 18/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 573ms/epoch - 8ms/step
Epoch 19/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 544ms/epoch - 7ms/step
Epoch 20/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 552ms/epoch - 7ms/step
```

```
Inp = Input(shape=(784,))
x = Dense(n_hidden_1, activation='relu', name = "Hidden_Layer_1")(Inp)
x = Dense(n_hidden_2, activation='relu', name = "Hidden_Layer_2")(x)
x = Dense(n_hidden_3, activation='relu', name = "Hidden_Layer_3")(x)
x = Dense(n_hidden_4, activation='relu', name = "Hidden_Layer_4")(x)
output = Dense(num_digits, activation='softmax', name = "Output_Layer")(x)

learning_rate = 0.01
adam = keras.optimizers.Adam(lr=learning_rate)
model2a = Model(Inp, output)

model2a.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
history2a = model2a.fit(X_train, y_train,
                        batch_size = batch_size,
                        epochs = training_epochs,
                        verbose = 2,
                        validation_data=(X_cv, y_cv))
```

```
Epoch 1/20
74/74 - 3s - loss: nan - accuracy: 0.6411 - val_loss: nan - val_accuracy: 0.0911 - 3s/epoch - 40ms/step
Epoch 2/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 673ms/epoch - 9ms/step
Epoch 3/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 683ms/epoch - 9ms/step
Epoch 4/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 806ms/epoch - 11ms/step
Epoch 5/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 1s/epoch - 14ms/step
Epoch 6/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 1s/epoch - 15ms/step
Epoch 7/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 751ms/epoch - 10ms/step
Epoch 8/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 680ms/epoch - 9ms/step
Epoch 9/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 767ms/epoch - 10ms/step
Epoch 10/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 725ms/epoch - 10ms/step
Epoch 11/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 685ms/epoch - 9ms/step
```

```
Epoch 12/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 664ms/epoch - 9ms/step
Epoch 13/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 731ms/epoch - 10ms/step
Epoch 14/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 742ms/epoch - 10ms/step
Epoch 15/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 671ms/epoch - 9ms/step
Epoch 16/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 638ms/epoch - 9ms/step
Epoch 17/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 650ms/epoch - 9ms/step
Epoch 18/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 714ms/epoch - 10ms/step
Epoch 19/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 660ms/epoch - 9ms/step
Epoch 20/20
74/74 - 1s - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911 - 719ms/epoch - 10ms/step
```

```
Inp = Input(shape=(784,))
x = Dense(n_hidden_1, activation='relu', name = "Hidden_Layer_1")(Inp)
x = Dense(n_hidden_2, activation='relu', name = "Hidden_Layer_2")(x)
x = Dense(n_hidden_3, activation='relu', name = "Hidden_Layer_3")(x)
x = Dense(n_hidden_4, activation='relu', name = "Hidden_Layer_4")(x)
output = Dense(num_digits, activation='softmax', name = "Output_Layer")(x)

learning_rate = 0.5
adam = keras.optimizers.Adam(lr=learning_rate)
model2b = Model(Inp, output)

model2b.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
history2b = model2b.fit(X_train, y_train,
                        batch_size = batch_size,
                        epochs = training_epochs,
                            validation_data=(X_cv, y_cv))
```

```
Epoch 1/20
74/74 [==============================] - 3s 13ms/step - loss: nan - accuracy: 0.5564 - val_loss: nan - val_accuracy: 0.0911
Epoch 2/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 3/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 4/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 5/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 6/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 7/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 8/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 9/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 10/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 11/20
74/74 [==============================] - 1s 15ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
```

```
Epoch 12/20
74/74 [==============================] - 1s 16ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 13/20
74/74 [==============================] - 1s 14ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 14/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 15/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 16/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 17/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 18/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 19/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 20/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
```

```
n_input = 784
n_hidden_1 = 300
n_hidden_2 = 100
n_hidden_3 = 100
n_hidden_4 = 100
n_hidden_5 = 200
num_digits = 10
```

```
Inp = Input(shape=(784,))
x = Dense(n_hidden_1, activation='relu', name = "Hidden_Layer_1")(Inp)
x = Dense(n_hidden_2, activation='relu', name = "Hidden_Layer_2")(x)
x = Dense(n_hidden_3, activation='relu', name = "Hidden_Layer_3")(x)
x = Dense(n_hidden_4, activation='relu', name = "Hidden_Layer_4")(x)
x = Dense(n_hidden_5, activation='relu', name = "Hidden_Layer_5")(x)
output = Dense(num_digits, activation='softmax', name = "Output_Layer")(x)
```

```
model3 = Model(Inp, output)
model3.summary()
```

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 784)]             0

 Hidden_Layer_1 (Dense)      (None, 300)               235500

 Hidden_Layer_2 (Dense)      (None, 100)               30100

 Hidden_Layer_3 (Dense)      (None, 100)               10100

 Hidden_Layer_4 (Dense)      (None, 100)               10100

 Hidden_Layer_5 (Dense)      (None, 200)               20200

 Output_Layer (Dense)        (None, 10)                2010

=================================================================
Total params: 308,010
Trainable params: 308,010
Non-trainable params: 0
_________________________________________________________________
```

```
adam = keras.optimizers.Adam(lr=0.01)

model3.compile(loss='categorical_crossentropy',
               optimizer='adam',
               metrics=['accuracy'])
```

```
history3 = model3.fit(X_train, y_train,
                      batch_size = batch_size,
                      epochs = training_epochs,
                      validation_data=(X_cv, y_cv))
```

```
Epoch 1/20
74/74 [==============================] - 3s 21ms/step - loss: nan - accuracy: 0.1130 - val_loss: nan - val_accuracy: 0.0911
Epoch 2/20
74/74 [==============================] - 1s 14ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 3/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 4/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 5/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 6/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 7/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 8/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 9/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 10/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 11/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911

Epoch 12/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 13/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 14/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 15/20
74/74 [==============================] - 1s 15ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 16/20
74/74 [==============================] - 1s 17ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 17/20
74/74 [==============================] - 1s 13ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 18/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 19/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 20/20
74/74 [==============================] - 1s 10ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
```

```
n_input = 784
n_hidden_1 = 300
n_hidden_2 = 100
n_hidden_3 = 100
n_hidden_4 = 200
num_digits = 10
```

```
Inp = Input(shape=(784,))
x = Dense(n_hidden_1, activation='relu', name = "Hidden_Layer_1")(Inp)
x = Dropout(0.3)(x)
x = Dense(n_hidden_2, activation='relu', name = "Hidden_Layer_2")(x)
x = Dropout(0.3)(x)
x = Dense(n_hidden_3, activation='relu', name = "Hidden_Layer_3")(x)
x = Dropout(0.3)(x)
x = Dense(n_hidden_4, activation='relu', name = "Hidden_Layer_4")(x)
output = Dense(num_digits, activation='softmax', name = "Output_Layer")(x)
```

```
model4 = Model(Inp, output)
model4.summary()
```

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 784)]             0

 Hidden_Layer_1 (Dense)      (None, 300)               235500

 dropout (Dropout)           (None, 300)               0

 Hidden_Layer_2 (Dense)      (None, 100)               30100

 dropout_1 (Dropout)         (None, 100)               0

 Hidden_Layer_3 (Dense)      (None, 100)               10100

 dropout_2 (Dropout)         (None, 100)               0

 Hidden_Layer_4 (Dense)      (None, 200)               20200

 Output_Layer (Dense)        (None, 10)                2010

Total params: 297,910
Trainable params: 297,910
Non-trainable params: 0
```

```
model4.compile(loss='categorical_crossentropy',
               optimizer='adam',
               metrics=['accuracy'])
```

```
history = model4.fit(X_train, y_train,
                     batch_size = batch_size,
                     epochs = training_epochs,
                     validation_data=(X_cv, y_cv))
```

```
Epoch 1/20
74/74 [==============================] - 2s 15ms/step - loss: nan - accuracy: 0.1227 - val_loss: nan - val_accuracy: 0.0911
Epoch 2/20
74/74 [==============================] - 1s 13ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 3/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 4/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 5/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 6/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 7/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 8/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 9/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 10/20
74/74 [==============================] - 1s 14ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 11/20
74/74 [==============================] - 1s 17ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 12/20
74/74 [==============================] - 1s 17ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 13/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 14/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 15/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 16/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 17/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 18/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 19/20
74/74 [==============================] - 1s 12ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
Epoch 20/20
74/74 [==============================] - 1s 11ms/step - loss: nan - accuracy: 0.1022 - val_loss: nan - val_accuracy: 0.0911
```

```
test_pred = pd.DataFrame(model4.predict(X_test, batch_size=200))
test_pred = pd.DataFrame(test_pred.idxmax(axis = 1))
test_pred.index.name = 'ImageId'
test_pred = test_pred.rename(columns = {0: 'Label'}).reset_index()
test_pred['ImageId'] = test_pred['ImageId'] + 1

test_pred.head()
```

```
64/64 [==============================] - 0s 5ms/step
```

	ImageId	Label
0	1	NaN
1	2	NaN
2	3	NaN
3	4	NaN
4	5	NaN