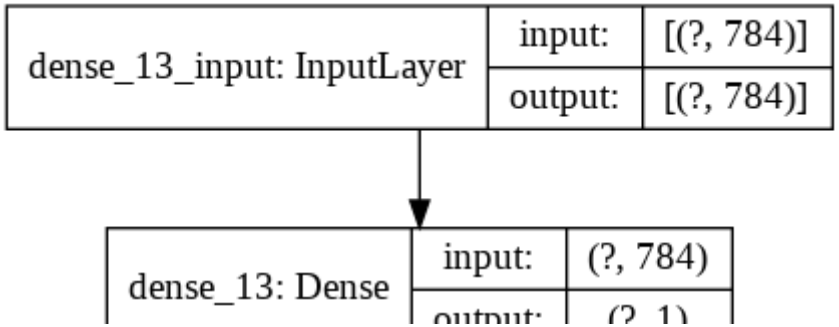
```
import numpy as np
from tensorflow import keras
print (keras.__version__)
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
x_train_final = x_train_new.reshape((-1, 784))
x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
x_test_final = x_test_new.reshape((-1, 784))
x_train_final = x_train_final / 255
x_test_final = x_test_final / 255


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras




model = Sequential()

model.add(Dense(1, input_shape=(784,)))
model.add( Activation('sigmoid'))
# this is the output for one-hot vector
#model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)



model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
```

```
  y_train_new,
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
```

```
2.4.0
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 1)                 785
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 785
Trainable params: 785
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0786 - binary_accuracy: 0.9897
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0207 - binary_accuracy: 0.9976
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0150 - binary_accuracy: 0.9976
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0124 - binary_accuracy: 0.9977
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0108 - binary_accuracy: 0.9979
67/67 [==============================] - 0s 2ms/step - loss: 0.0066 - binary_accuracy: 0.9995
```

```
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```




```
import numpy as np
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
```

```
# load dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
# count the number of unique train labels
unique, counts = np.unique(y_train, return_counts=True)
print("Train labels: ", dict(zip(unique, counts)))
```

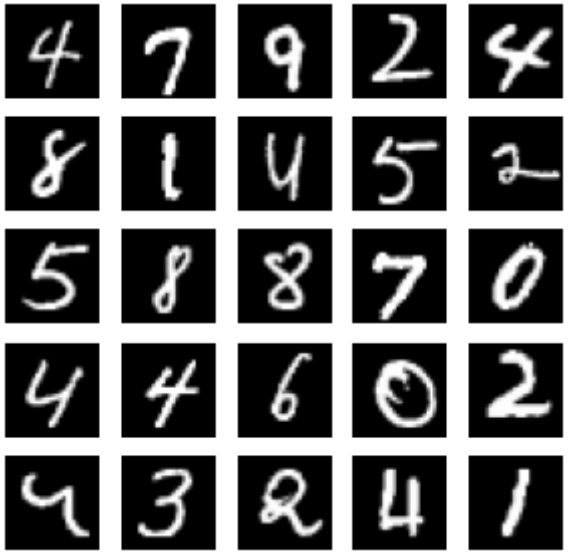
```
    Train labels:  {0: 5923, 1: 6742, 2: 5958, 3: 6131, 4: 5842, 5: 5421, 6: 5918, 7: 6265, 8: 5851, 9: 5949}
```

```
# count the number of unique test labels
unique, counts = np.unique(y_test, return_counts=True)
print("Test labels: ", dict(zip(unique, counts)))
```

```
    Test labels:  {0: 980, 1: 1135, 2: 1032, 3: 1010, 4: 982, 5: 892, 6: 958, 7: 1028, 8: 974, 9: 1009}
```

```
# sample 25 mnist digits from train dataset
indexes = np.random.randint(0, x_train.shape[0], size=25)
images = x_train[indexes]
```

```
labels = y_train[indexes]


# plot the 25 mnist digits
plt.figure(figsize=(5,5))
for i in range(len(indexes)):
    plt.subplot(5, 5, i + 1)
    image = images[i]
    plt.imshow(image, cmap='gray')
    plt.axis('off')
```



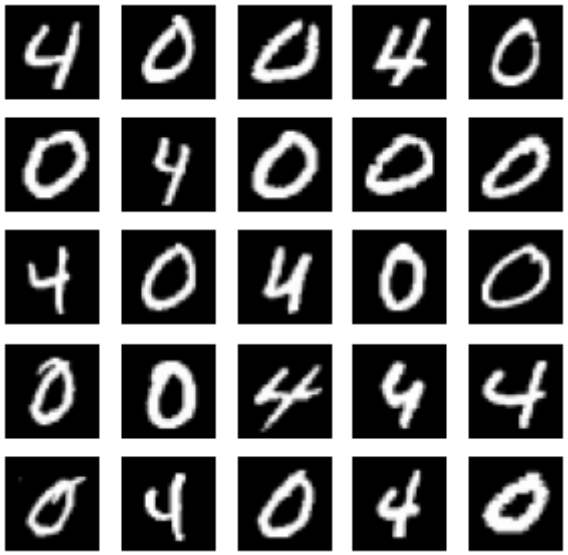
```
plt.savefig("mnist-samples.png")
plt.show()
plt.close('all')
```

```
<Figure size 432x288 with 0 Axes>
```

```
# load mnist dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()


#print(f'{x_train}')
```

```
#print(f'{x_train}')

X_train = x_train
Y_train = y_train
X_test = x_test
Y_test = y_test




import numpy as np

train_filter = np.where((y_train == 0 ) | (y_train == 4))
test_filter = np.where((y_test == 0) | (y_test == 4))
x_train, y_train = x_train[train_filter], y_train[train_filter]
x_test, y_test = x_test[test_filter], y_test[test_filter]

#print(f'{x_train[0]}')

print('{train_filter}')

     {train_filter}




# sample 25 mnist digits from train dataset
indexes = np.random.randint(0, x_train.shape[0], size=25)
images = x_train[indexes]
labels = y_train[indexes]


# plot the 25 mnist digits
plt.figure(figsize=(5,5))
for i in range(len(indexes)):
    plt.subplot(5, 5, i + 1)
    image = images[i]
```

```
image = images[1]
plt.imshow(image, cmap='gray')
plt.axis('off')
```

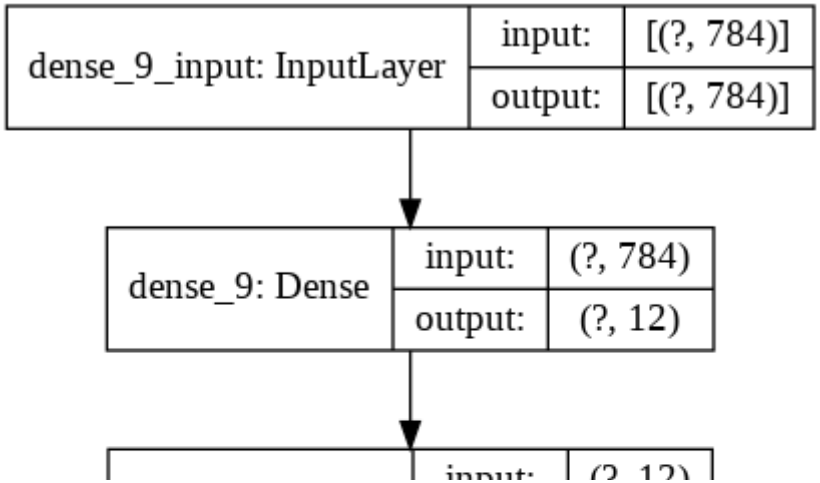


minst mlp processing

```
import numpy as np
from tensorflow import keras
print (keras.__version__)
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
x_train_final = x_train_new.reshape((-1, 784))
x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
x_test_final = x_test_new.reshape((-1, 784))
x_train_final = x_train_final / 255
x_test_final = x_test_final / 255


from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras


model = Sequential()
model.add(Dense(12, input_shape=(784,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)



model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
2.4.0
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 12)                9420
_________________________________________________________________
dense_10 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 9
=================================================================
Total params: 9,533
Trainable params: 9,533
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.1036 - binary_accuracy: 0.9840
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0096 - binary_accuracy: 0.9979
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0065 - binary_accuracy: 0.9984
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0053 - binary_accuracy: 0.9987
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0046 - binary_accuracy: 0.9986
67/67 [==============================] - 0s 2ms/step - loss: 0.0023 - binary_accuracy: 0.9995
```




```
# validate the model on test dataset to determine generalization
```

```
_, acc = model.evaluate(x_test_final,
                        y_test_new,
                        batch_size=128,
                        verbose=0)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
```

```
    Test accuracy: 100.0%
```

```
import numpy as np
from tensorflow import keras
print (keras.__version__)
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
x_train_final = x_train_new.reshape((-1, 784))
x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
x_test_final = x_test_new.reshape((-1, 784))
x_train_final = x_train_final / 255
x_test_final = x_test_final / 255
```

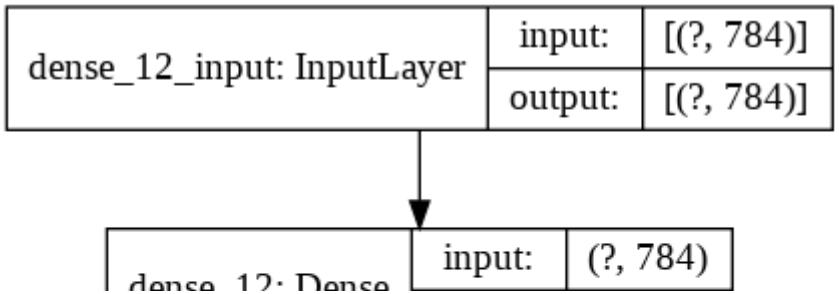
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras
```

```
model = Sequential()
model.add(Dense(12, input_shape=(784,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
2.4.0
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 12)                9420
_________________________________________________________________
dense_13 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_14 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 5
=================================================================
Total params: 9,565
Trainable params: 9,565
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.1052 - binary_accuracy: 0.9853
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0081 - binary_accuracy: 0.9979
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0053 - binary_accuracy: 0.9985
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0042 - binary_accuracy: 0.9989
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0036 - binary_accuracy: 0.9991
67/67 [==============================] - 0s 2ms/step - loss: 0.0020 - binary_accuracy: 0.9995
```

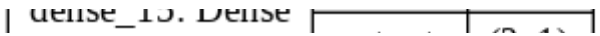


Double-click (or enter) to edit

```
# validate the model on test dataset to determine generalization
```

```
_, acc = model.evaluate(x_test_final,
                        y_test_new,
                        batch_size=128,
                        verbose=0)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
```

```
Test accuracy: 100.0%
```



Double-click (or enter) to edit

1. Task 1: Here you need to use MLP for classification. (a) Use the pair of classes, of MNIST data used by you in assignment 5, with around 6000 training patterns per class. So, you will have 12000 training patterns totally. Train an MLP each with 1, 2, 3 and 4 hidden layers using Backpropa- gation. (b) Take 2000 test patterns (1000 per class) and classify using the networks trained in (a). Compute the classification accuracy on the test patterns. (c) Pick the best MLP based on (a) and (b) and vary the following: i. Number of nodes in the hidden layers. ii. Tanh, Relu, and Logistic activation functions iii. Hyperparameters: Momentum term, Early stopping, and Learning Rate (d) Report the classification accuracies obtained and analyse.

```
import numpy
from sklearn.model_selection import GridSearchCV
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
# Function to create model, required for KerasClassifier
def create_model():
```

```
  model = Sequential()
  model.add(Dense(12, input_shape=(784,), activation='relu'))
  model.add(Dense(8, activation='relu'))
  model.add(Dense(4, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model

model = KerasClassifier(build_fn=create_model, verbose=0)
# define the grid search parameters
batch_size = [10,20]
epochs = [10, 50]
#optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
learn_rate = [0.001, 0.01, 0.1, 0.2, 0.3]
momentum = [0.0, 0.2, 0.4, 0.6, 0.8, 0.9]


param_grid = dict(batch_size=batch_size, epochs=epochs)
#grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
#param_grid = dict(optimizer=optimizer)



from sklearn.model_selection import RandomizedSearchCV
parameters ={'batch_size':[5,10],
             'nb_epoch':[10,20,50],
             #'optimizer':['adam','rmsprop','SGD'],
             #'kernel_initializer':['random_uniform'],
             #'units':[4,8,13]
              }
#random_search= RandomizedSearchCV(estimator=model, param_distributions=parameters,n_iter=20,n_jobs=-1,cv=5)

grid = GridSearchCV(estimator=model, param_grid=parameters, n_jobs=-1, cv=3)
X = x_train_final
Y=y_train_new
grid_result = grid.fit(X, Y)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
```

```
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.998816 using {'batch_size': 10, 'nb_epoch': 10}
0.998026 (0.000487) with: {'batch_size': 5, 'nb_epoch': 10}
0.997552 (0.000487) with: {'batch_size': 5, 'nb_epoch': 20}
0.998737 (0.000805) with: {'batch_size': 5, 'nb_epoch': 50}
0.998816 (0.000670) with: {'batch_size': 10, 'nb_epoch': 10}
0.997947 (0.001116) with: {'batch_size': 10, 'nb_epoch': 20}
0.998421 (0.000487) with: {'batch_size': 10, 'nb_epoch': 50}
```

```
# i. Number of nodes in the hidden layers.
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py:281: UserWarning: The total space o
  % (grid_size, self.n_iter, grid_size), UserWarning)
Best score obtained: -0.20594636561788607
Parameters:
```

```
# i. Number of nodes in the hidden layers.

node_list = [ 12,24,36,48,60]
for node in node_list :

  import numpy as np
  from tensorflow import keras
  print (keras.__version__)
  from keras.datasets import mnist
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
  x_train_final = x_train_new.reshape((-1, 784))
  x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
  x_test_final = x_test_new.reshape((-1, 784))
  x_train_final, x_train_final = x_train_final / 255
```

```
x_train_final = x_train_final / 255
x_test_final = x_test_final / 255


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras

print(f" Processing - Node changes { node}")
model = Sequential()
model.add(Dense(node , input_shape=(784,), activation='relu'))
model.add(Dense(node -4, activation='relu'))
model.add(Dense(node - 8, activation='relu'))
model.add(Dense( 1, activation='sigmoid'))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)







model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
2.4.0
 Processing - Node changes 12
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 12)                9420
_________________________________________________________________
dense_5 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 5
=================================================================
Total params: 9,565
Trainable params: 9,565
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.1548 - binary_accuracy: 0.9641
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0086 - binary_accuracy: 0.9979
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0055 - binary_accuracy: 0.9985
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0044 - binary_accuracy: 0.9987
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0038 - binary_accuracy: 0.9989
67/67 [==============================] - 0s 2ms/step - loss: 0.0017 - binary_accuracy: 0.9995
2.4.0
 Processing - Node changes 24
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 24)                18840
_________________________________________________________________
dense_9 (Dense)              (None, 20)                500
_________________________________________________________________
dense_10 (Dense)             (None, 16)                336
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 17
=================================================================
```

```
Total params: 19,693
Trainable params: 19,693
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0605 - binary_accuracy: 0.9957
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0070 - binary_accuracy: 0.9984
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0050 - binary_accuracy: 0.9985
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0041 - binary_accuracy: 0.9987
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0035 - binary_accuracy: 0.9990
67/67 [==============================] - 0s 2ms/step - loss: 0.0019 - binary_accuracy: 0.9995
```

New Section

```
# Tanh, Relu, and Logistic activation functions.

fn_list = [ "tanh", "relu","sigmoid"]
for fn in fn_list :

  import numpy as np
  from tensorflow import keras
  print (keras.__version__)
  from keras.datasets import mnist
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
  x_train_final = x_train_new.reshape((-1, 784))
  x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
  x_test_final = x_test_new.reshape((-1, 784))
  x_train_final = x_train_final / 255
  x_test_final = x_test_final / 255
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras

print(f" Processing - activation function  changes { fn }")
model = Sequential()
node = 16 # No. of nodes - fixed
model.add(Dense(node , input_shape=(784,), activation=fn))
model.add(Dense(node -4, activation=fn))
model.add(Dense(node - 8, activation=fn))
model.add(Dense( 1, activation=fn))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```



```
model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
   2.4.0
    Processing - activation function  changes tanh
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_24 (Dense)             (None, 16)                12560
_________________________________________________________________
dense_25 (Dense)             (None, 12)                204
_________________________________________________________________
dense_26 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0238 - binary_accuracy: 0.9962
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0102 - binary_accuracy: 0.9990
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0094 - binary_accuracy: 0.9987
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0088 - binary_accuracy: 0.9990
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0079 - binary_accuracy: 0.9992
67/67 [==============================] - 0s 2ms/step - loss: 0.0103 - binary_accuracy: 0.9981
2.4.0
 Processing - activation function  changes relu
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, 16)                12560
_________________________________________________________________
dense_29 (Dense)             (None, 12)                204
_________________________________________________________________
dense_30 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
```

```
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0297 - binary_accuracy: 0.9931
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0116 - binary_accuracy: 0.9987
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0060 - binary_accuracy: 0.9984
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0056 - binary_accuracy: 0.9991
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0047 - binary_accuracy: 0.9988
67/67 [==============================] - 0s 2ms/step - loss: 0.0100 - binary_accuracy: 0.9991
```

```
#Hyperparameters: Momentum term, Early stopping, and Learning Rate

from keras import backend as K


learning_rate_list = [ 0.001, 0.010, 0.1, 1]

for l_r in  learning_rate_list:

  import numpy as np
  from tensorflow import keras
  print (keras.__version__)
  from keras.datasets import mnist
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
  x_train_final = x_train_new.reshape((-1, 784))
  x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
  x_test_final = x_test_new.reshape((-1, 784))
  x_train_final = x_train_final / 255
  x_test_final = x_test_final / 255
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras

print(f" Processing - learning rate  changes { l_r }")
model = Sequential()
node = 16 # No. of nodes - fixed

model.add(Dense(node , input_shape=(784,), activation=fn))
model.add(Dense(node -4, activation=fn))
model.add(Dense(node - 8, activation=fn))
model.add(Dense( 1, activation=fn))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)




print (f'model {model}')


keras.optimizers.Adam(lr=l_r)

#K.set_value(model.optimizer.learning_rate, l_r)



model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True,
  epochs=5
```

```
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
2.4.0
 Processing - learning rate  changes 0.1
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_84 (Dense)             (None, 16)                12560
_________________________________________________________________
dense_85 (Dense)             (None, 12)                204
_________________________________________________________________
dense_86 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_87 (Dense)             (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f0851a3dac8>
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.2745 - binary_accuracy: 0.9721
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0388 - binary_accuracy: 0.9988
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0166 - binary_accuracy: 0.9992
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0096 - binary_accuracy: 0.9994
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0070 - binary_accuracy: 0.9993
67/67 [==============================] - 0s 2ms/step - loss: 0.0070 - binary_accuracy: 0.9991
2.4.0
 Processing - learning rate  changes 1
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_88 (Dense)             (None, 16)                12560
```

```
_________________________________________________________________
dense_89 (Dense)             (None, 12)                204       
_________________________________________________________________
dense_90 (Dense)             (None, 8)                 104       
_________________________________________________________________
dense_91 (Dense)             (None, 1)                 9         
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f0852b86d30>
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.2457 - binary_accuracy: 0.9464
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0348 - binary_accuracy: 0.9987
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0152 - binary_accuracy: 0.9991
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0088 - binary_accuracy: 0.9995
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0068 - binary_accuracy: 0.9993
```

```
from keras import backend as K


learning_rate_list = [ 0.001, 0.010, 0.1, 1]

for l_r in  learning_rate_list:

  import numpy as np
  from tensorflow import keras
  print (keras.__version__)
  from keras.datasets import mnist
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
  x_train_final = x_train_new.reshape((-1, 784))
  x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
  x_test_final = x_test_new.reshape((-1, 784))
  x_train_final = x_train_final / 255
  x_test_final = x_test_final / 255
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras

print(f" Processing - learning rate  changes { l_r }")
model = Sequential()
node = 16 # No. of nodes - fixed

model.add(Dense(node , input_shape=(784,), activation=fn))
model.add(Dense(node -4, activation=fn))
model.add(Dense(node - 8, activation=fn))
model.add(Dense( 1, activation=fn))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)




print (f'model {model}')


opt = keras.optimizers.Adam(lr=l_r)

#K.set_value(model.optimizer.learning_rate, l_r)



model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True
```

```
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
2.4.0
 Processing - learning rate  changes 0.001
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_116 (Dense)            (None, 16)                12560
_________________________________________________________________
dense_117 (Dense)            (None, 12)                204
_________________________________________________________________
dense_118 (Dense)            (None, 8)                 104
_________________________________________________________________
dense_119 (Dense)            (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f08529e1cc0>
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.2486 - binary_accuracy: 0.9589
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0343 - binary_accuracy: 0.9990
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0144 - binary_accuracy: 0.9994
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0093 - binary_accuracy: 0.9991
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0062 - binary_accuracy: 0.9994
67/67 [==============================] - 0s 2ms/step - loss: 0.0052 - binary_accuracy: 0.9995
2.4.0
 Processing - learning rate  changes 0.01
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense_120 (Dense)            (None, 16)                12560
_________________________________________________________________
dense_121 (Dense)            (None, 12)                204
_________________________________________________________________
dense_122 (Dense)            (None, 8)                 104
_________________________________________________________________
dense_123 (Dense)            (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f085e4f3208>
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0495 - binary_accuracy: 0.9952
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0072 - binary_accuracy: 0.9987
Epoch 3/5
792/792 [==============================] - 2s 3ms/step - loss: 0.0055 - binary_accuracy: 0.9985
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0050 - binary_accuracy: 0.9987
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0021 - binary_accuracy: 0.9993
```

```
from keras import backend as K


momentum_list = [ 0.0, 0.2, 0.4, 0.9]

for momentum in  momentum_list:

  import numpy as np
  from tensorflow import keras
  print (keras.__version__)
  from keras.datasets import mnist
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
  x_train_final = x_train_new.reshape((-1, 784))
  x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
  x_test_final = x_test_new.reshape((-1, 784))
  x_train_final = x_train_final / 255
```

```
x_test_final = x_test_final / 255


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras

print(f" Processing - momemtum  changes { momentum }")
model = Sequential()
node = 16 # No. of nodes - fixed

model.add(Dense(node , input_shape=(784,), activation=fn))
model.add(Dense(node -4, activation=fn))
model.add(Dense(node - 8, activation=fn))
model.add(Dense( 1, activation=fn))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)




print (f'model {model}')

l_r = 1
opt=keras.optimizers.SGD(lr=l_r, momentum=momentum)
#optimizer.momentum.set_value(0.04)

#K.set_value(model.optimizer.learning_rate, l_r)


#opt = keras.optimizers.Adam(momentum=0.9)
#model.compile(..., optimizer=opt)
```

```
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['binary_accuracy'])
model.fit(
  x=x_train_final,
  y=y_train_new,
  shuffle=True,
  epochs=5,
  batch_size=16
)
eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
2.4.0
 Processing - momemtum  changes 0.4
Model: "sequential_44"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_176 (Dense)            (None, 16)                12560
_________________________________________________________________
dense_177 (Dense)            (None, 12)                204
_________________________________________________________________
dense_178 (Dense)            (None, 8)                 104
_________________________________________________________________
dense_179 (Dense)            (None, 1)                 9

=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f08fdda52b0>
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0465 - binary_accuracy: 0.9758
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0068 - binary_accuracy: 0.9983
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0024 - binary_accuracy: 0.9994
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0027 - binary_accuracy: 0.9993
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0037 - binary_accuracy: 0.9989
67/67 [==============================] - 0s 2ms/step - loss: 0.0054 - binary_accuracy: 0.9981
2.4.0
```

```
2.4.0
 Processing - momemtum  changes 0.9
Model: "sequential_45"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_180 (Dense)            (None, 16)                12560
_________________________________________________________________
dense_181 (Dense)            (None, 12)                204
_________________________________________________________________
dense_182 (Dense)            (None, 8)                 104
_________________________________________________________________
dense_183 (Dense)            (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f0852c22d68>
Epoch 1/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0374 - binary_accuracy: 0.9848
Epoch 2/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0120 - binary_accuracy: 0.9968
Epoch 3/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0110 - binary_accuracy: 0.9969
Epoch 4/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0054 - binary_accuracy: 0.9981
Epoch 5/5
792/792 [==============================] - 2s 2ms/step - loss: 0.0042 - binary_accuracy: 0.9986
```

```
early_stopping_list = [ 'min','max','auto']

for e_s in  early_stopping_list:

  import numpy as np
  from tensorflow import keras
  print (keras.__version__)
  from keras.datasets import mnist
  (x_train, y_train), (x_test, y_test) = mnist.load_data()
  x_train_new, y_train_new = x_train[(y_train==0) | (y_train==1)], y_train[(y_train==0) | (y_train==1)]
  x_train_final = x_train_new.reshape((-1, 784))
```

```
x_test_new, y_test_new = x_test[(y_test==0) | (y_test==1)], y_test[(y_test==0) | (y_test==1)]
x_test_final = x_test_new.reshape((-1, 784))
x_train_final = x_train_final / 255
x_test_final = x_test_final / 255


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist

from tensorflow import keras as k



print(f" Processing - early stopping  changes { e_s }")
model = Sequential()
node = 16 # No. of nodes - fixed
fn ='sigmoid'

model.add(Dense(node , input_shape=(784,), activation=fn))
model.add(Dense(node -4, activation=fn))
model.add(Dense(node - 8, activation=fn))
model.add(Dense( 1, activation=fn))



model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)


print (f'model {model}')


callback = k.callbacks.EarlyStopping(
#monitor='binary_accuracy', min_delta=0.0001,
monitor='accuracy', min_delta=0.0001,

patience=1)
```

```
l_r = 1
momentum=0.9
opt=keras.optimizers.SGD(lr=l_r, momentum=momentum)
#optimizer.momentum.set_value(0.04)

#K.set_value(model.optimizer.learning_rate, l_r)


#opt = keras.optimizers.Adam(momentum=0.9)
#model.compile(..., optimizer=opt)

#model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['binary_accuracy'])

model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])


'''
history = model.fit(
          x=x_train_final,
          y=y_train_new,
          shuffle=True,
          epochs=10, batch_size=1, callbacks=[callback],verbose=0)
'''


#from keras.callbacks import EarlyStopping
#early_stopping_monitor = EarlyStopping(patience=2)
history = model.fit(x=x_train_final, y=y_train_new, validation_split=0.3, epochs=100, callbacks=[callback])

l = len(history.history['loss'])
print(f' No of epochs ran {l}')  # Only 4 epochs are run.


eval = model.evaluate(x=x_test_final, y=y_test_new)
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)
```

```
   2.4.0
    Processing - early stopping  changes max
   Model: "sequential_9"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_36 (Dense)             (None, 16)                12560
_________________________________________________________________
dense_37 (Dense)             (None, 12)                204
_________________________________________________________________
dense_38 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_39 (Dense)             (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f284e0d60b8>
Epoch 1/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0568 - accuracy: 0.9748 - val_loss: 0.0048 -
Epoch 2/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0074 - accuracy: 0.9984 - val_loss: 0.0045 -
Epoch 3/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0044 - accuracy: 0.9990 - val_loss: 0.0029 -
Epoch 4/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0066 - accuracy: 0.9984 - val_loss: 0.0029 -
 No of epochs ran 4
67/67 [==============================] - 0s 2ms/step - loss: 0.0025 - accuracy: 0.9991
2.4.0
 Processing - early stopping  changes auto
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_40 (Dense)             (None, 16)                12560
_________________________________________________________________
dense_41 (Dense)             (None, 12)                204
_________________________________________________________________
dense_42 (Dense)             (None, 8)                 104
_________________________________________________________________
dense_43 (Dense)             (None, 1)                 9
=================================================================
Total params: 12,877
Trainable params: 12,877
Non-trainable params: 0
```

```
_________________________________________________________________
model <tensorflow.python.keras.engine.sequential.Sequential object at 0x7f2842742e10>
Epoch 1/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0683 - accuracy: 0.9649 - val_loss: 0.0044 -
Epoch 2/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0113 - accuracy: 0.9966 - val_loss: 0.0040 -
Epoch 3/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0052 - accuracy: 0.9984 - val_loss: 0.0025 -
Epoch 4/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0047 - accuracy: 0.9992 - val_loss: 0.0035 -
Epoch 5/100
278/278 [==============================] - 1s 3ms/step - loss: 0.0044 - accuracy: 0.9985 - val_loss: 0.0028 -
```