Amirkabir
University of Technology
Tehran Polytechnic

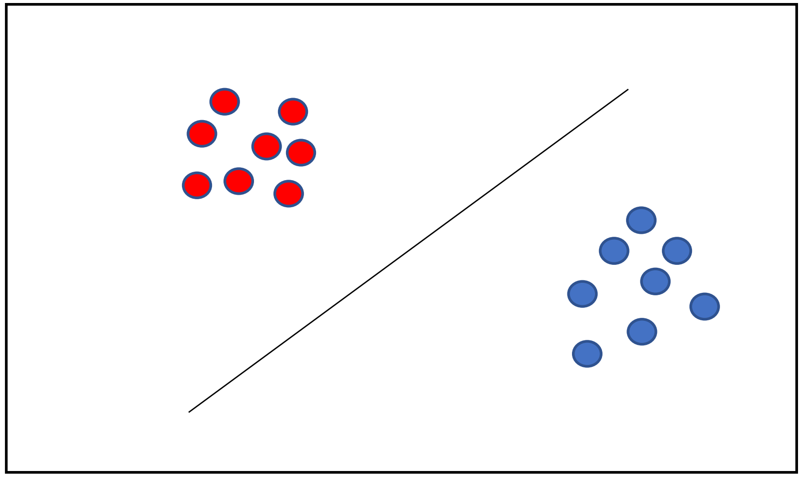
A Discriminative Feature Learning Approach For Deep Face Recognition

Professor : M. B. Menhaj

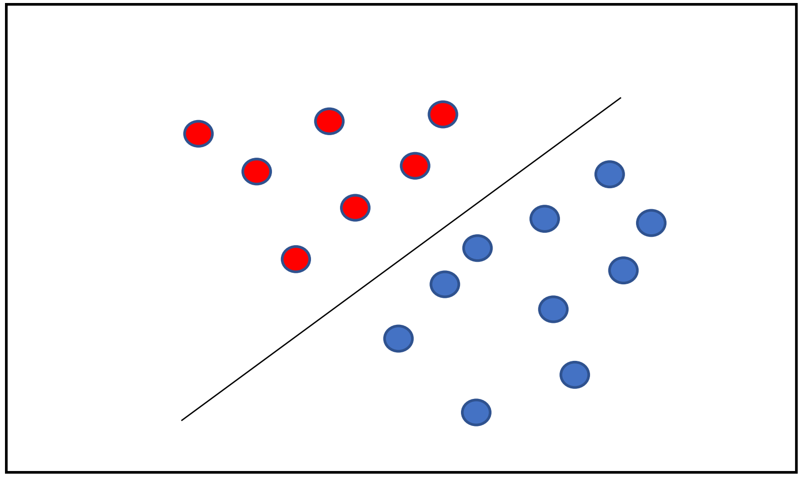
Mahdi Ardestani

رویکرد مقاله :

Discriminative Feature



Separable Feature

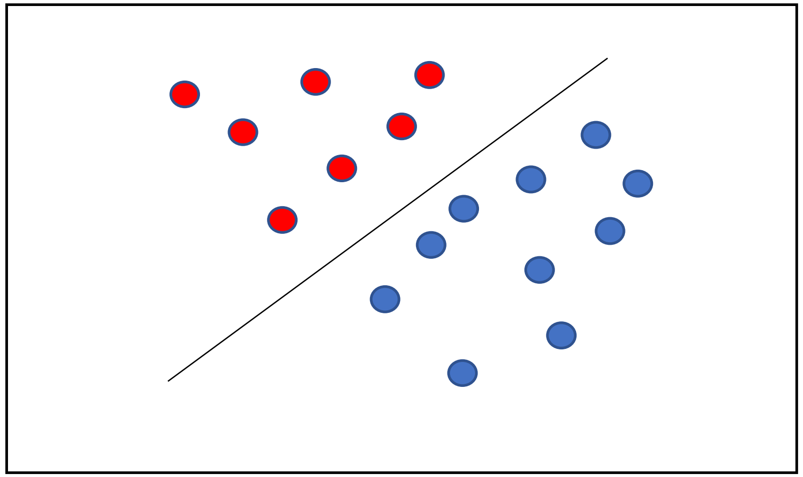


مقاله در مورد این رویکرد جداسازی به این نکته اشاره دارد:

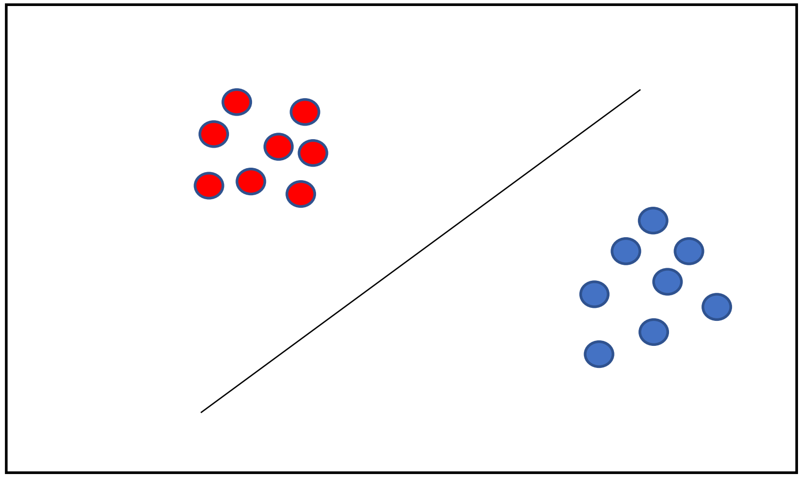
For face recognition task, the deeply learned features need to be not only separable but also discriminative

به همین علت softmax به تنهایی کافی نیست.

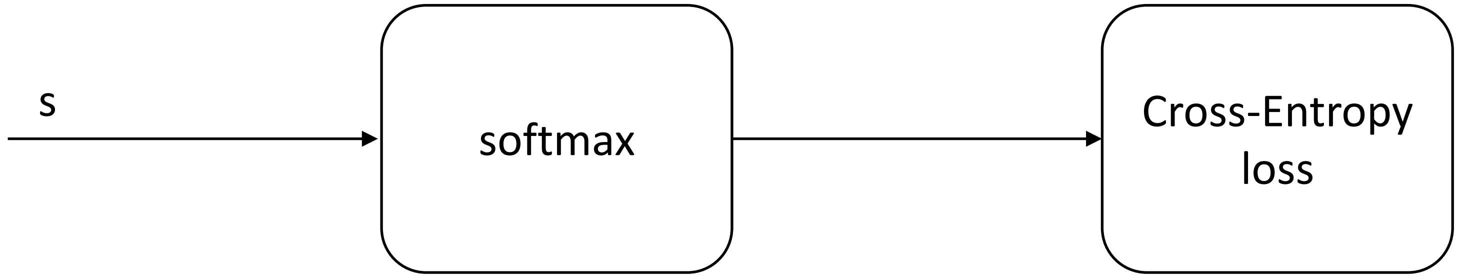
Softmax loss



Softmax loss + ?



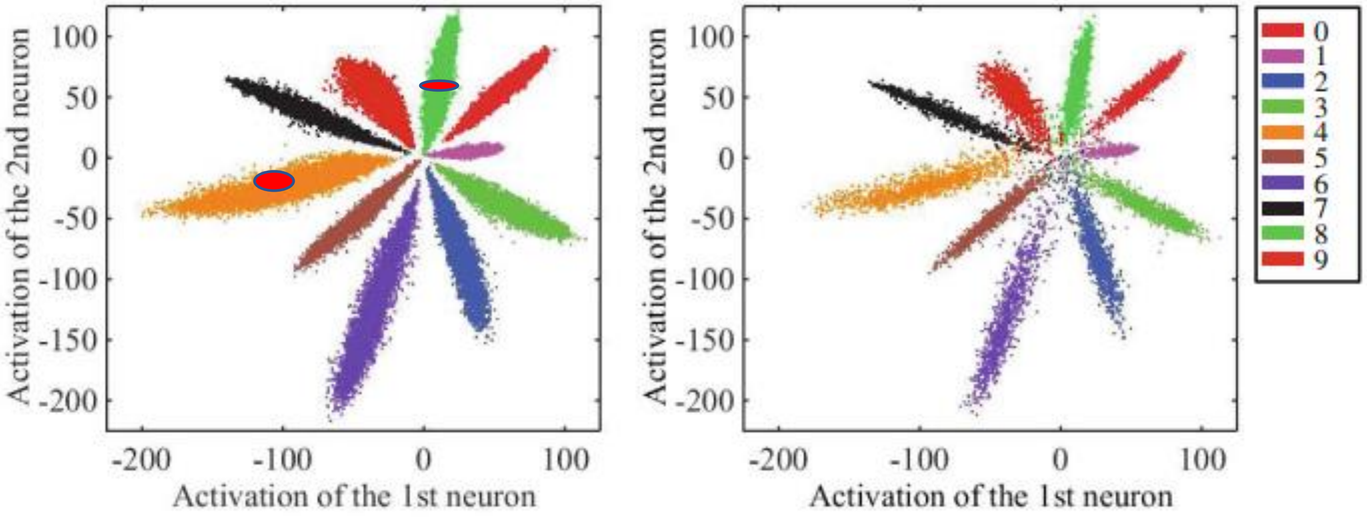
تعريف softmax loss:



$$L_s = -\sum_{i=1}^{m} \log \frac{e^{W_{yi}^{T} x_i + b_{yi}}}{\sum_{j=1}^{n} e^{W_{yi}^{T} x_i + b_{yi}}}$$

$x_i \in R^d$	*ith deep feature*
$W_j \in R^d$	*jth column of the weight*
$b \in R^n$	*bias term*
m	*size of* min−*batch*
n	*number of class*

نتیجه حاصل از اعمال softmax :



بنابراین هدفمان از پیاده سازی به صورت زیر تعریف می‌شود:

Minimizing the intra-class variations while keeping the features of different classes separable is the key

در نتیجه loss زیر را تعریف می‌کنیم:

$$L_c = \frac{1}{2}\sum_{i=1}^{m} \| x_i - c_{yi} \|_2^{\,2}$$

x_i:Deep feature

c_{yi}:Class center

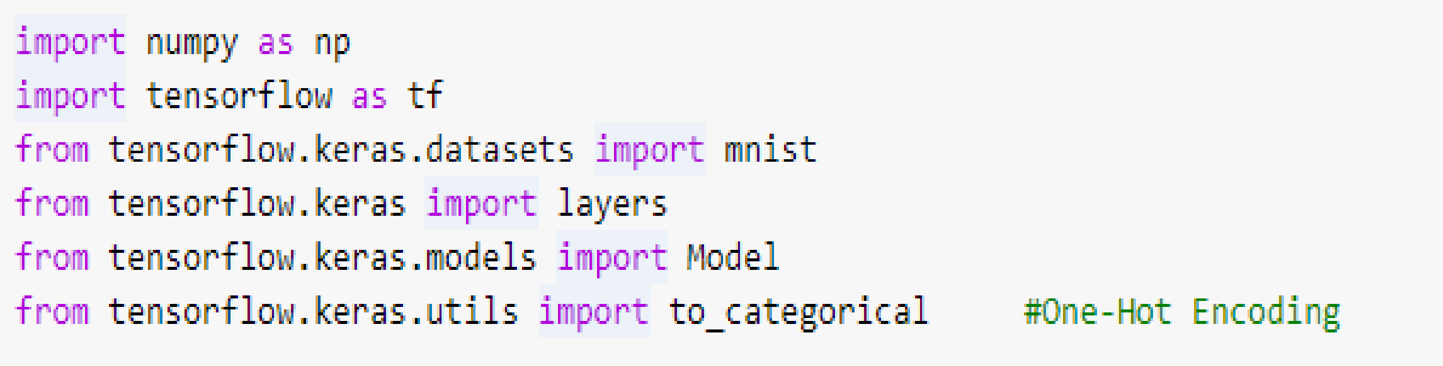
در نتیجه loss نهایی به صورت زیر تعریف می‌شود:

$$L = L_s + \lambda L_c$$

$$L = -\sum_{i=1}^{m} \log \frac{e^{W_{yi}{}^{T} x_i + b_{yi}}}{\sum_{j=1}^{n} e^{W_{yi}{}^{T} x_i + b_{yi}}} + \frac{\lambda}{2}\sum_{i=1}^{m} \| x_i - c_{yi} \|_2^2$$

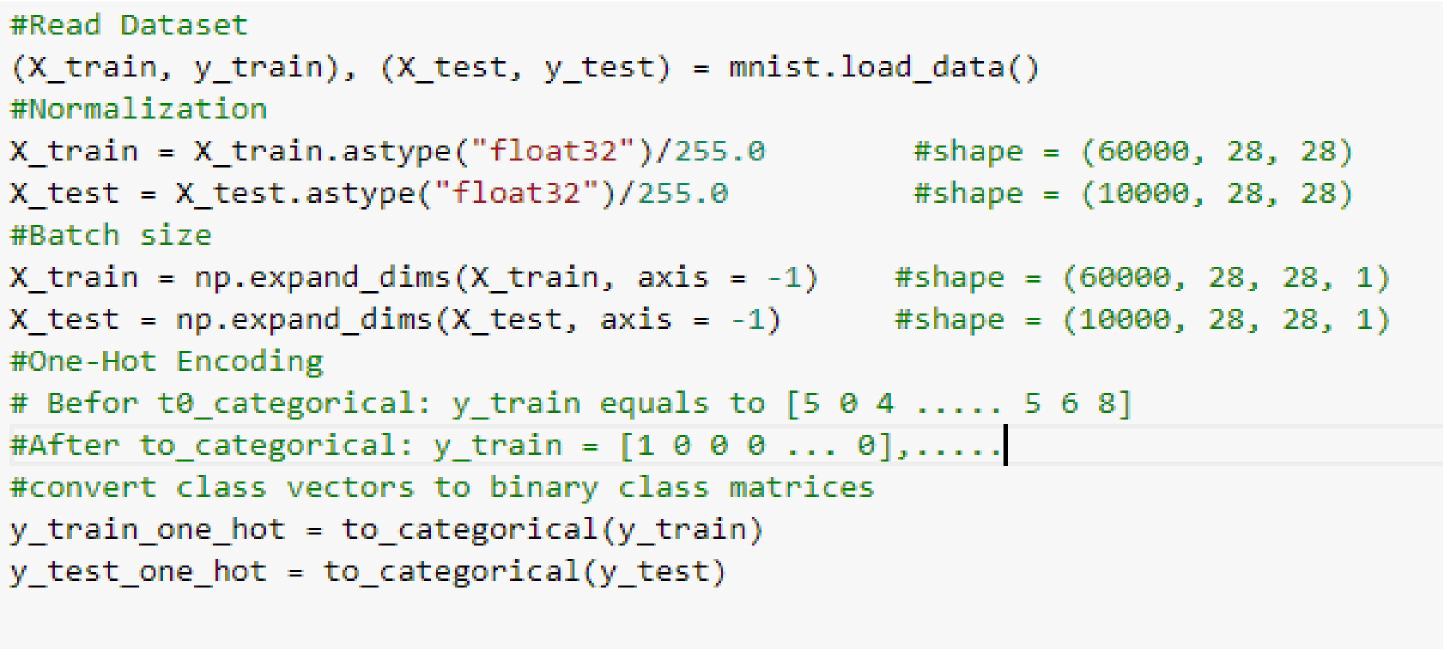
شبیه سازی:

اضافه کردن ماژول‌های مورد نیاز



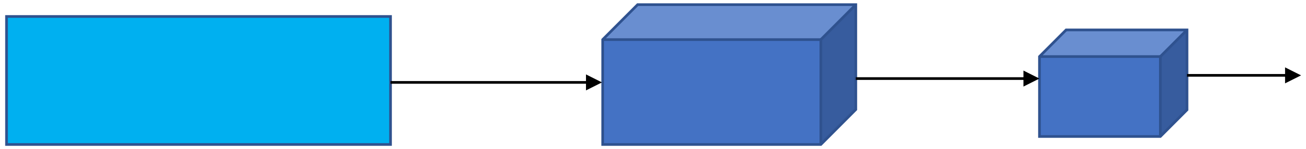
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras import layers
from tensorflow.keras.models import Model
from tensorflow.keras.utils import to_categorical     #One-Hot Encoding
```

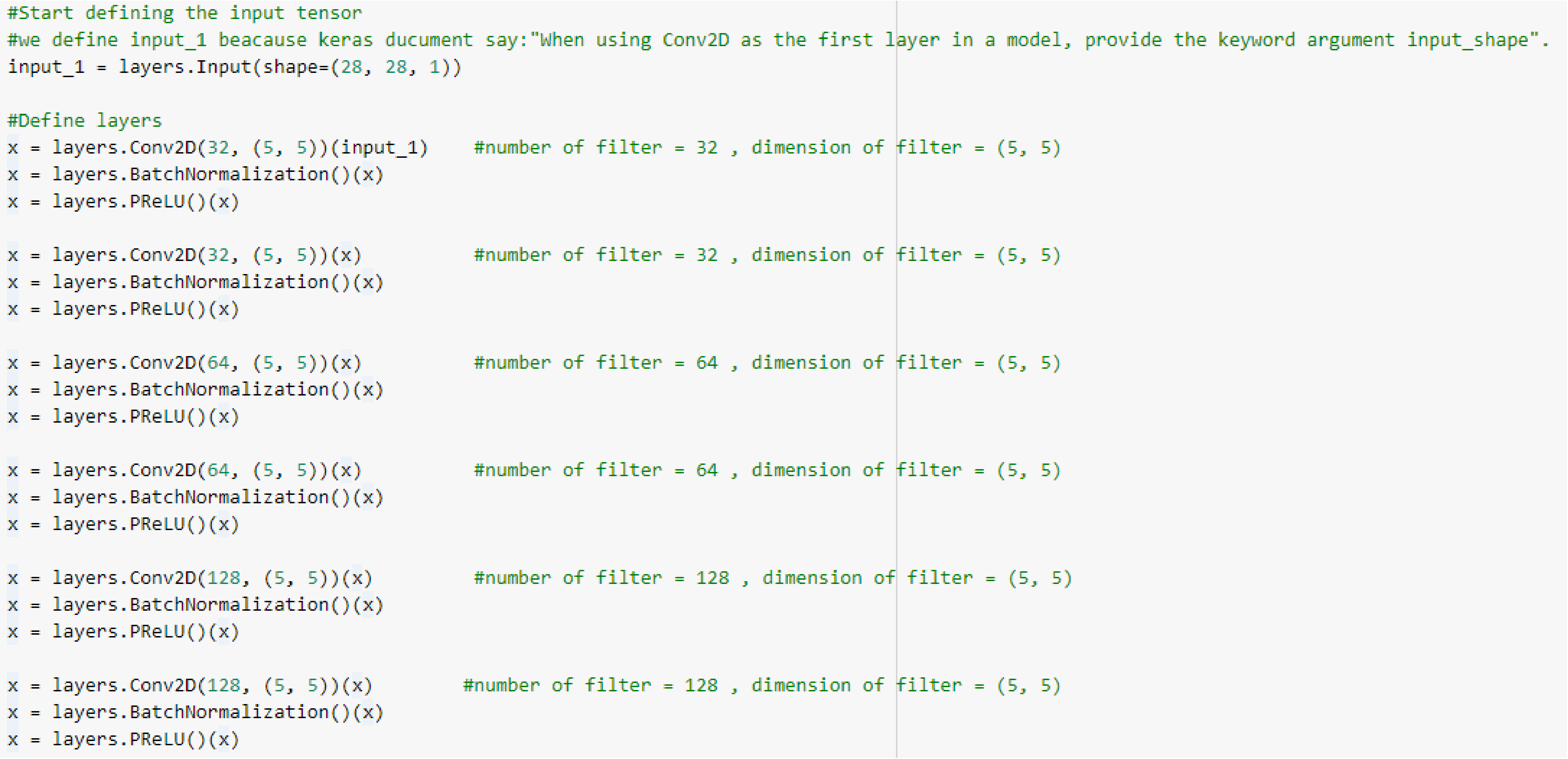
خواندن دیتاست و نرمالیزه کردن دیتا



```
#Read Dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
#Normalization
X_train = X_train.astype("float32")/255.0        #shape = (60000, 28, 28)
X_test = X_test.astype("float32")/255.0          #shape = (10000, 28, 28)
#Batch size
X_train = np.expand_dims(X_train, axis = -1)    #shape = (60000, 28, 28, 1)
X_test = np.expand_dims(X_test, axis = -1)      #shape = (10000, 28, 28, 1)
#One-Hot Encoding
# Befor t0_categorical: y_train equals to [5 0 4 ..... 5 6 8]
#After to_categorical: y_train = [1 0 0 0 ... 0],.....
#convert class vectors to binary class matrices
y_train_one_hot = to_categorical(y_train)
y_test_one_hot = to_categorical(y_test)
```

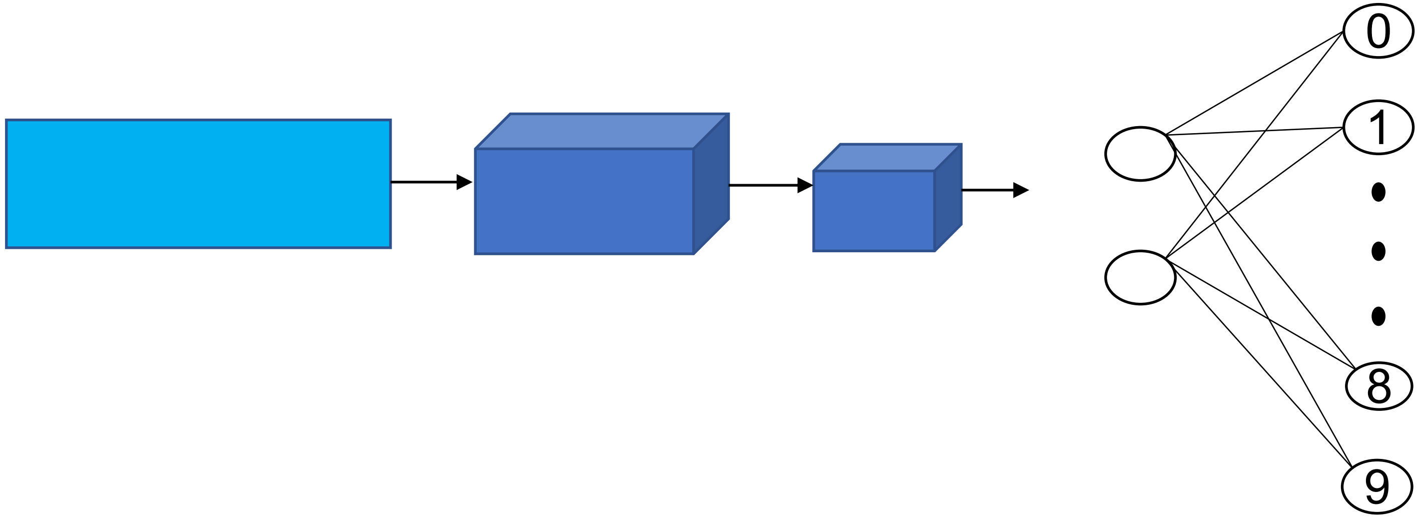
ساخت قسمت اصلی شبکه:



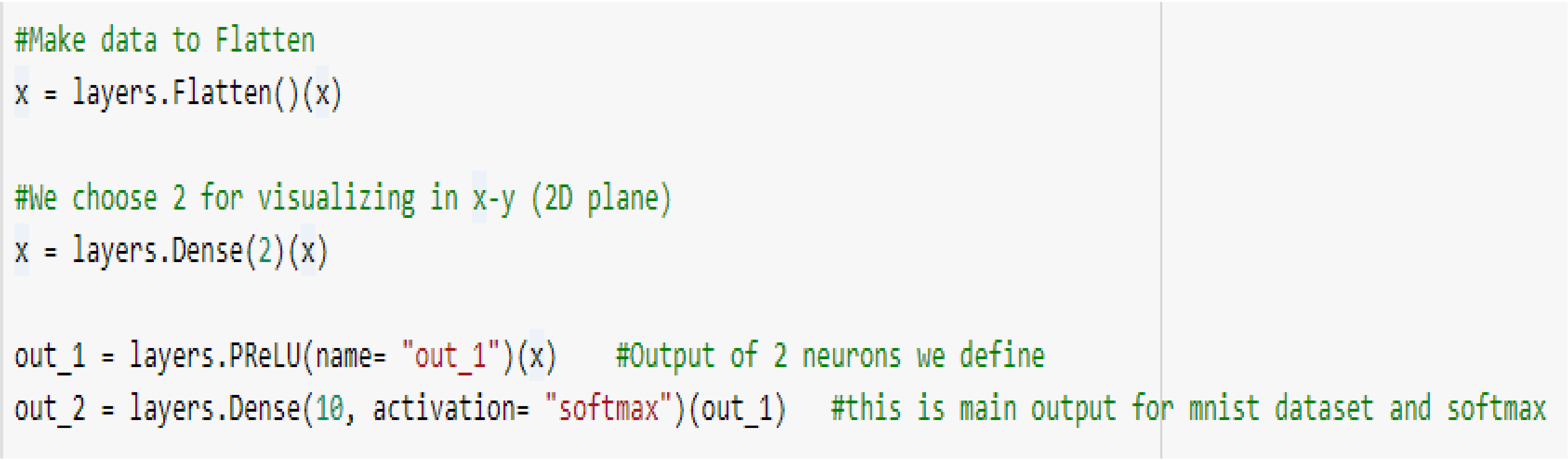


```
#Start defining the input tensor
#we define input_1 beacause keras ducument say:"When using Conv2D as the first layer in a model, provide the keyword argument input_shape".
input_1 = layers.Input(shape=(28, 28, 1))

#Define layers
x = layers.Conv2D(32, (5, 5))(input_1)    #number of filter = 32 , dimension of filter = (5, 5)
x = layers.BatchNormalization()(x)
x = layers.PReLU()(x)

x = layers.Conv2D(32, (5, 5))(x)          #number of filter = 32 , dimension of filter = (5, 5)
x = layers.BatchNormalization()(x)
x = layers.PReLU()(x)

x = layers.Conv2D(64, (5, 5))(x)          #number of filter = 64 , dimension of filter = (5, 5)
x = layers.BatchNormalization()(x)
x = layers.PReLU()(x)

x = layers.Conv2D(64, (5, 5))(x)          #number of filter = 64 , dimension of filter = (5, 5)
x = layers.BatchNormalization()(x)
x = layers.PReLU()(x)

x = layers.Conv2D(128, (5, 5))(x)         #number of filter = 128 , dimension of filter = (5, 5)
x = layers.BatchNormalization()(x)
x = layers.PReLU()(x)

x = layers.Conv2D(128, (5, 5))(x)        #number of filter = 128 , dimension of filter = (5, 5)
x = layers.BatchNormalization()(x)
x = layers.PReLU()(x)
```

تکمیل قسمت اصلی:

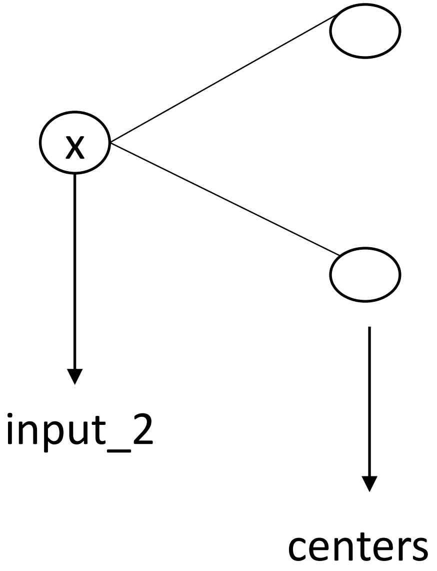


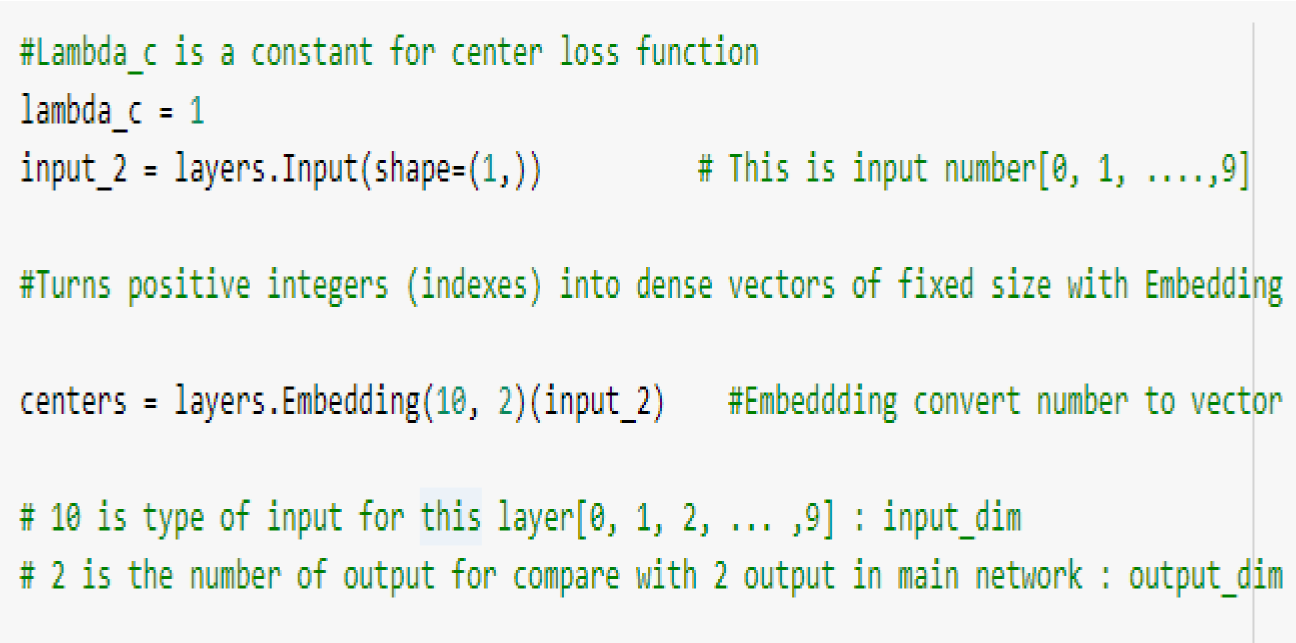
Softmax loss



```
#Make data to Flatten
x = layers.Flatten()(x)

#We choose 2 for visualizing in x-y (2D plane)
x = layers.Dense(2)(x)

out_1 = layers.PReLU(name= "out_1")(x)    #Output of 2 neurons we define
out_2 = layers.Dense(10, activation= "softmax")(out_1)   #this is main output for mnist dataset and softmax
```

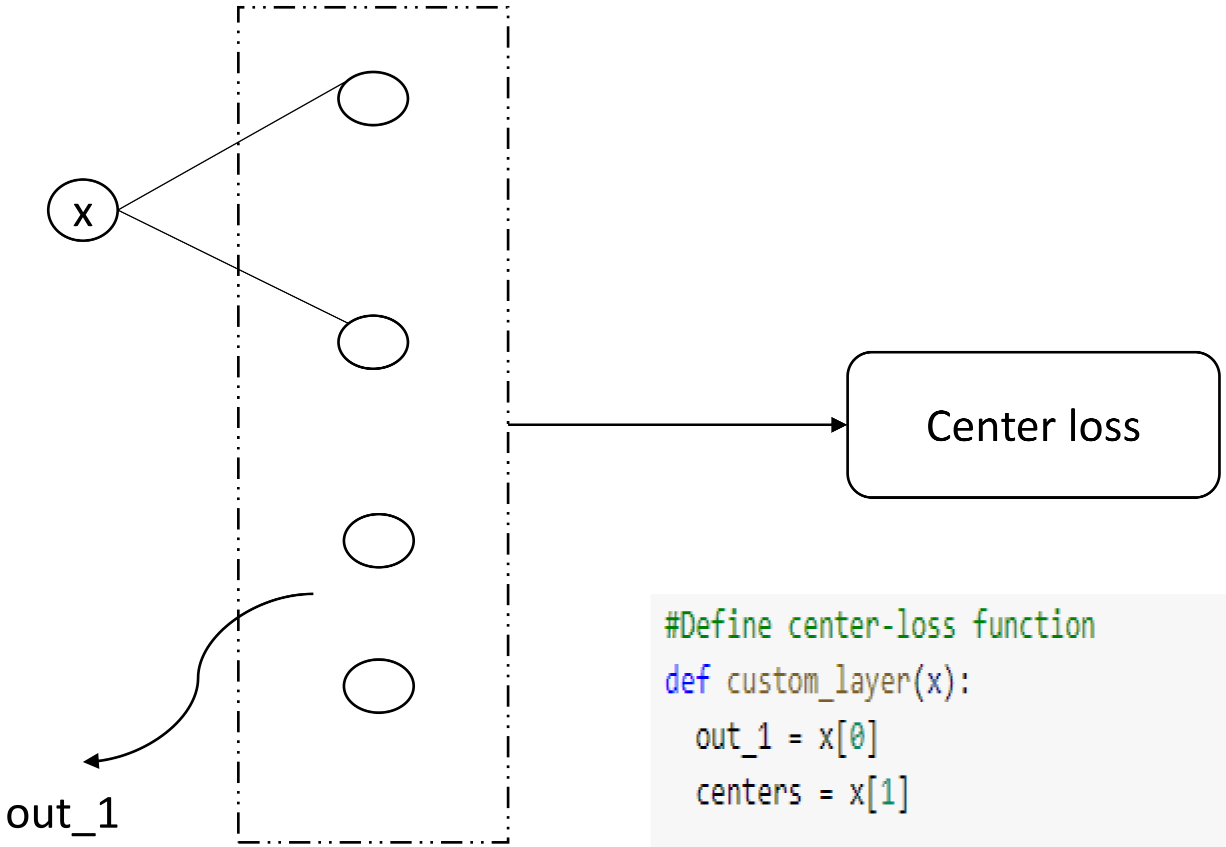
ساخت بخش دوم شبکه :

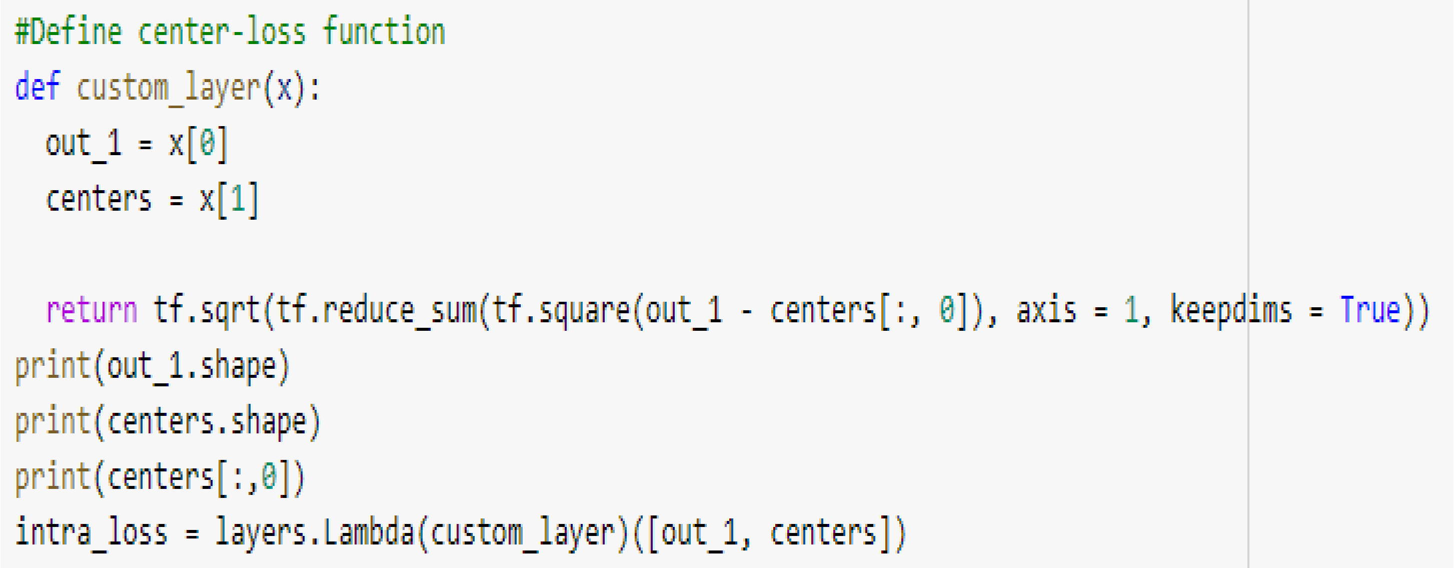




```
#Lambda_c is a constant for center loss function
lambda_c = 1
input_2 = layers.Input(shape=(1,))          # This is input number[0, 1, ....,9]

#Turns positive integers (indexes) into dense vectors of fixed size with Embedding


centers = layers.Embedding(10, 2)(input_2)    #Embeddding convert number to vector


# 10 is type of input for this layer[0, 1, 2, ... ,9] : input_dim
# 2 is the number of output for compare with 2 output in main network : output_dim
```

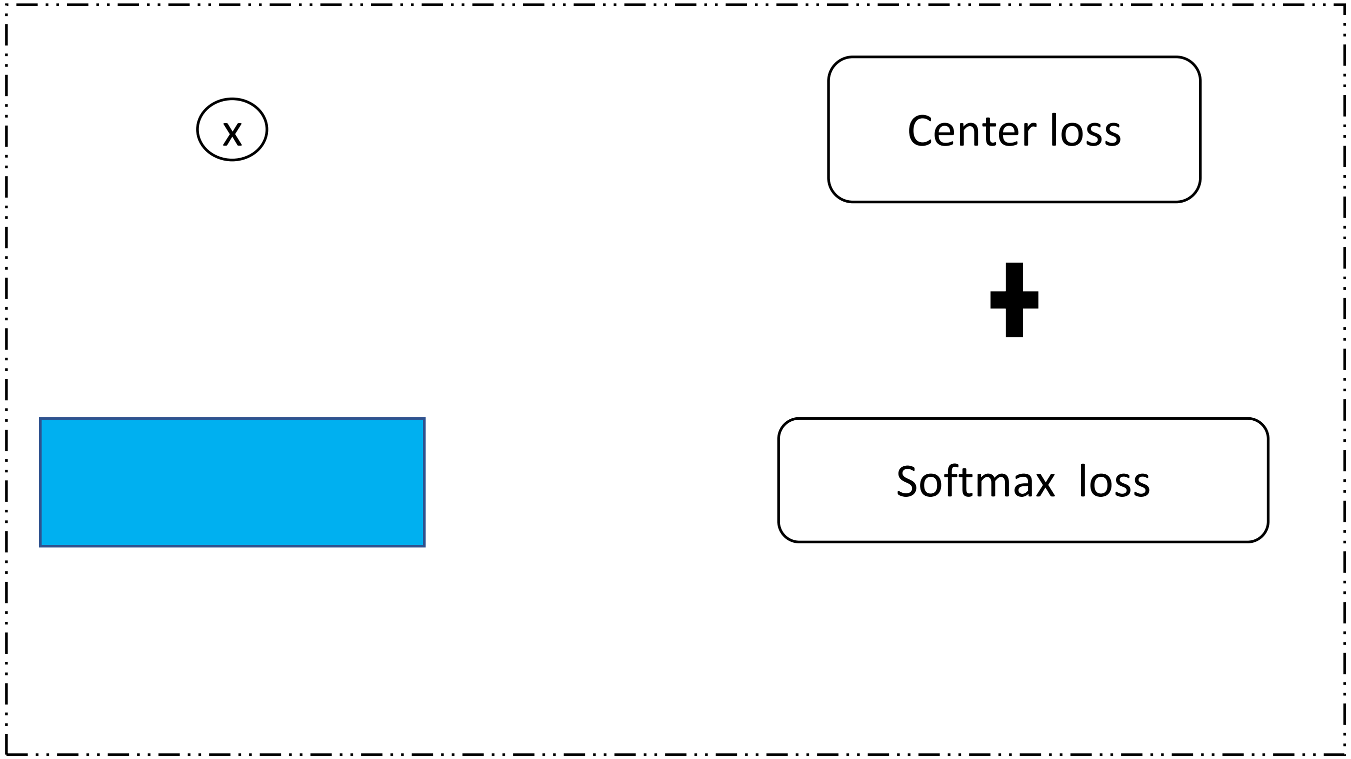
تعريف center loss:

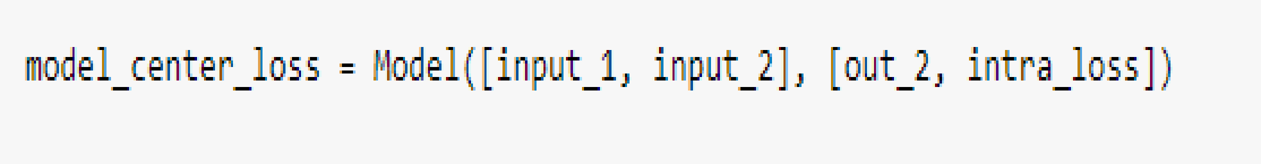




```
#Define center-loss function
def custom_layer(x):
  out_1 = x[0]
  centers = x[1]

  return tf.sqrt(tf.reduce_sum(tf.square(out_1 - centers[:, 0]), axis = 1, keepdims = True))
print(out_1.shape)
print(centers.shape)
print(centers[:,0])
intra_loss = layers.Lambda(custom_layer)([out_1, centers])
```

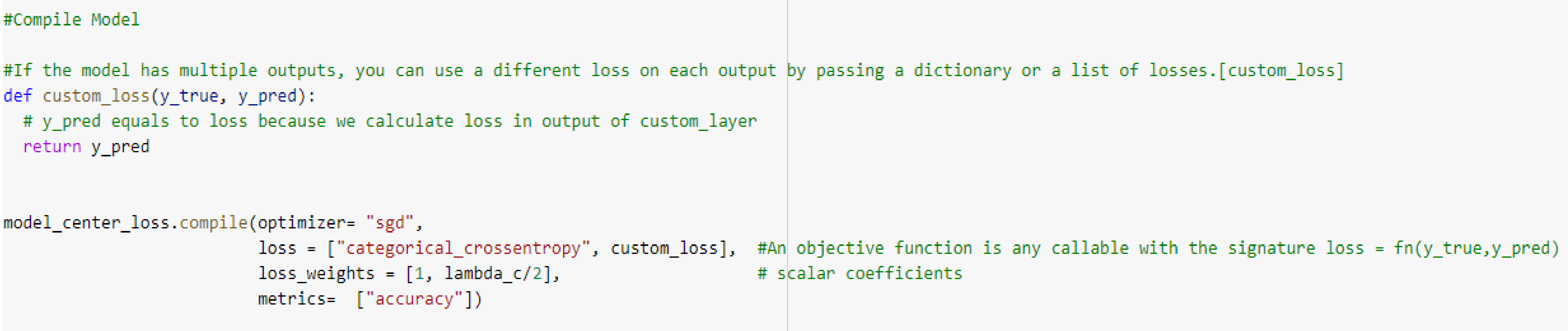
تشکیل کامل مدل:





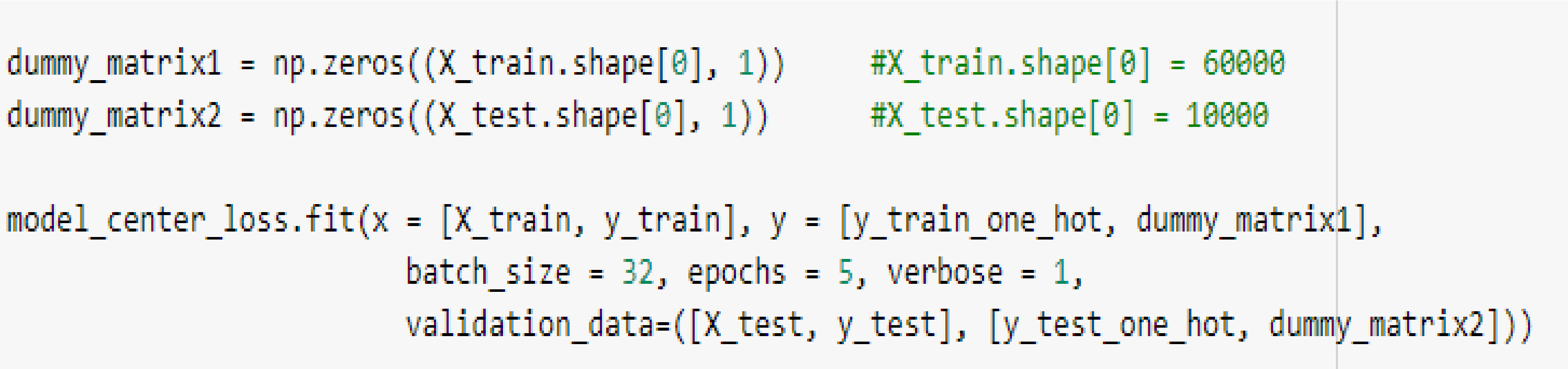
```
model_center_loss = Model([input_1, input_2], [out_2, intra_loss])
```

Compile کردن مدل



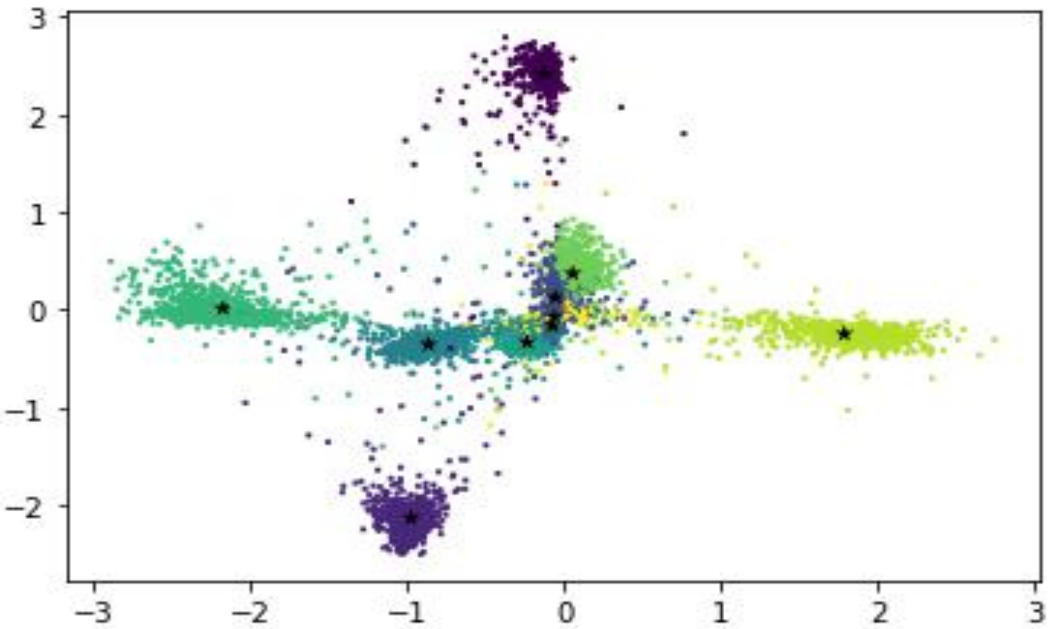
```
#Compile Model

#If the model has multiple outputs, you can use a different loss on each output by passing a dictionary or a list of losses.[custom_loss]
def custom_loss(y_true, y_pred):
  # y_pred equals to loss because we calculate loss in output of custom_layer
  return y_pred


model_center_loss.compile(optimizer= "sgd",
                          loss = ["categorical_crossentropy", custom_loss],  #An objective function is any callable with the signature loss = fn(y_true,y_pred)
                          loss_weights = [1, lambda_c/2],                    # scalar coefficients
                          metrics=  ["accuracy"])
```

Train شبکه



```
dummy_matrix1 = np.zeros((X_train.shape[0], 1))     #X_train.shape[0] = 60000
dummy_matrix2 = np.zeros((X_test.shape[0], 1))      #X_test.shape[0] = 10000

model_center_loss.fit(x = [X_train, y_train], y = [y_train_one_hot, dummy_matrix1],
                        batch_size = 32, epochs = 5, verbose = 1,
                        validation_data=([X_test, y_test], [y_test_one_hot, dummy_matrix2]))
```

نتیجه خروجی بعد از 5 ایپاک


3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3