MSDS 422 Week 9 Assignment 9: Autoencoder

Build an autoencoder for the Kaggle MNIST training dataset.

(1) Import and preprocess data

(2) Set parameters and define autoencoder backpropagation function

(3) Run a process to train autoencoder

(4) Visualise what the autoencoder has learned

Take the output of that autoencoder and use it to classify the MNIST test set observations. Submit your results to Kaggle.com.

(1) Import and preprocess data

In [39]:
```
import keras
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import gzip
%matplotlib inline
from keras.models import Model
from keras.optimizers import RMSprop
from keras.layers import Input,Dense,Flatten,Dropout,merge,Reshape,Conv2D,MaxPooling2D,UpSampling2D,Conv2DTranspose
from keras.layers.normalization import BatchNormalization
from keras.models import Model,Sequential
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adadelta, RMSprop,SGD,Adam
from keras import regularizers
from keras import backend as K
from keras.utils import to_categorical
import tensorflow as tf
```

In [40]:
```
df= pd.read_csv("Mtrain.csv")
df_test= pd.read_csv("Mtest.csv")
```

In [41]:
```
y = df['label'].values
X = df.drop(['label'], axis=1).values.reshape(-1, 28, 28,1)/255.

train_Y_one_hot = keras.utils.to_categorical(y)

X_test = df_test.values.reshape(-1, 28, 28,1)/255.

from sklearn.model_selection import train_test_split
train_X,valid_X,train_ground,valid_ground = train_test_split(X, X, test_size=0.2, random_state=13)

batch_size = 64
epochs = 1
inChannel = 1
x, y = 28, 28
input_img = tf.keras.Input(shape = (x, y, inChannel))
num_classes = 10
```

(2) Set parameters and define autoencoder backpropagation function

In [42]:
```
def encoder(input_img):
    conv1 = keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same')(input_img) #28 x 28 x 32
    conv1 = keras.layers.BatchNormalization()(conv1)
    conv1 = keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
    conv1 = keras.layers.BatchNormalization()(conv1)
    pool1 = keras.layers.MaxPool2D(pool_size=(2, 2))(conv1) #14 x 14 x 32
    conv2 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(pool1) #14 x 14 x 64
    conv2 = keras.layers.BatchNormalization()(conv2)
    conv2 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    conv2 = keras.layers.BatchNormalization()(conv2)
    pool2 = keras.layers.MaxPool2D(pool_size=(2, 2))(conv2) #7 x 7 x 64
    conv3 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(pool2) #7 x 7 x 128 (small and thick)
    conv3 = keras.layers.BatchNormalization()(conv3)
    conv3 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
    conv3 = keras.layers.BatchNormalization()(conv3)
    conv4 = keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(conv3) #7 x 7 x 256 (small and thick)
    conv4 = keras.layers.BatchNormalization()(conv4)
    conv4 = keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
    conv4 = keras.layers.BatchNormalization()(conv4)
    return conv4

def decoder(conv4):
    conv5 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(conv4) #7 x 7 x 128
    conv5 = keras.layers.BatchNormalization()(conv5)
    conv5 = keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(conv5)
    conv6 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(conv5) #7 x 7 x 64
    conv6 = keras.layers.BatchNormalization()(conv6)
    conv6 = keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(conv6)
    conv6 = keras.layers.BatchNormalization()(conv6)
    up1 = keras.layers.UpSampling2D((2,2))(conv6) #14 x 14 x 64
    conv7 = keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same')(up1) # 14 x 14 x 32
    conv7 = keras.layers.BatchNormalization()(conv7)
    conv7 = keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same')(conv7)
    conv7 = keras.layers.BatchNormalization()(conv7)
    up2 = keras.layers.UpSampling2D((2,2))(conv7) # 28 x 28 x 32
    decoded = keras.layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(up2) # 28 x 28 x 1
    return decoded
```

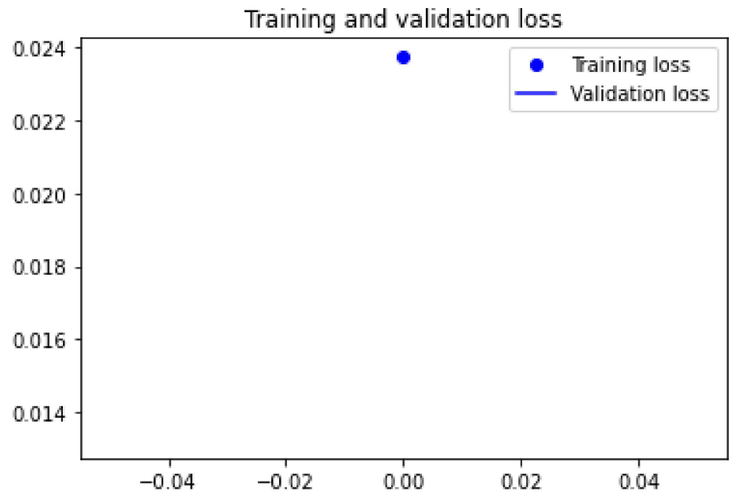
(3) Run a process to train autoencoder

In [43]:
```
autoencoder = keras.Model(input_img, decoder(encoder(input_img)))
autoencoder.compile(loss='mean_squared_error', optimizer = keras.optimizers.RMSprop())

#Train the model
autoencoder_train = autoencoder.fit(train_X, train_ground, batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(valid_X, valid_ground))

#Take a look at the loss
loss = autoencoder_train.history['loss']
val_loss = autoencoder_train.history['val_loss']
epochs = range(1)
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()

#Save the model
autoencoder.save_weights('autoencoder.h5')

# Predict with the encoder weights you just trained
# Change the labels from categorical to one-hot encoding
##train_Y_one_hot = keras.utils.to_categorical(train_labels)
##test_Y_one_hot = keras.utils.to_categorical(test_labels)

# Display the change for category label using one-hot encoding
##print('Original label:', train_labels[0])
##print('After conversion to one-hot:', train_Y_one_hot[0])

train_X,valid_X,train_label,valid_label = train_test_split(X,train_Y_one_hot,test_size=0.2,random_state=13)

#Check the shape
train_X.shape,valid_X.shape,train_label.shape,valid_label.shape

#fully connected layers
def fc(enco):
    flat = tf.keras.layers.Flatten()(enco)
    den = tf.keras.layers.Dense(128, activation='relu')(flat)
    out = tf.keras.layers.Dense(num_classes, activation='softmax')(den)
    return out

encode = encoder(input_img)
full_model = keras.Model(input_img,fc(encode))
```

Model: "functional_15"

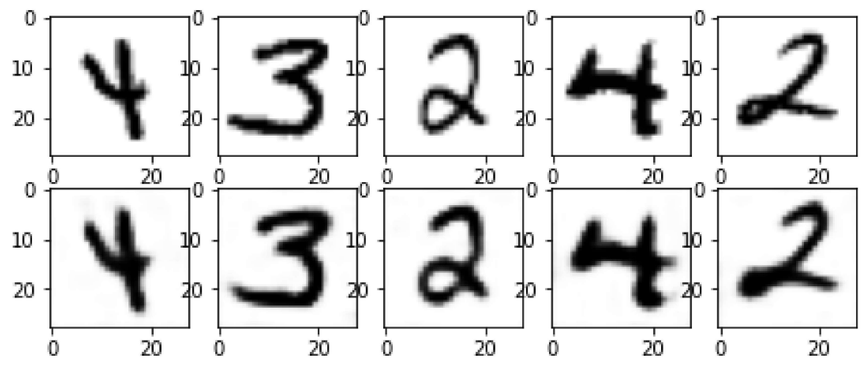
Layer (type)	Output Shape	Param #
input_10 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_64 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_60 (Batc	(None, 28, 28, 32)	128
conv2d_65 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_61 (Batc	(None, 28, 28, 32)	128
max_pooling2d_13 (MaxPooling	(None, 14, 14, 32)	0
conv2d_66 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_62 (Batc	(None, 14, 14, 64)	256
conv2d_67 (Conv2D)	(None, 14, 14, 64)	36928
batch_normalization_63 (Batc	(None, 14, 14, 64)	256
max_pooling2d_14 (MaxPooling	(None, 7, 7, 64)	0
conv2d_68 (Conv2D)	(None, 7, 7, 128)	73856
batch_normalization_64 (Batc	(None, 7, 7, 128)	512
conv2d_69 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_65 (Batc	(None, 7, 7, 128)	512
conv2d_70 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_66 (Batc	(None, 7, 7, 256)	1024
conv2d_71 (Conv2D)	(None, 7, 7, 256)	590080
batch_normalization_67 (Batc	(None, 7, 7, 256)	1024
conv2d_72 (Conv2D)	(None, 7, 7, 128)	295040
batch_normalization_68 (Batc	(None, 7, 7, 128)	512
conv2d_73 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_69 (Batc	(None, 7, 7, 128)	512

```
conv2d_74 (Conv2D)           (None, 7, 7, 64)          73792
_________________________________________________________________
batch_normalization_70 (Batc (None, 7, 7, 64)          256
_________________________________________________________________
conv2d_75 (Conv2D)           (None, 7, 7, 64)          36928
_________________________________________________________________
batch_normalization_71 (Batc (None, 7, 7, 64)          256
_________________________________________________________________
up_sampling2d_11 (UpSampling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_76 (Conv2D)           (None, 14, 14, 32)        18464
_________________________________________________________________
batch_normalization_72 (Batc (None, 14, 14, 32)        128
_________________________________________________________________
conv2d_77 (Conv2D)           (None, 14, 14, 32)        9248
_________________________________________________________________
batch_normalization_73 (Batc (None, 14, 14, 32)        128
_________________________________________________________________
up_sampling2d_12 (UpSampling (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_78 (Conv2D)           (None, 28, 28, 1)         289
=================================================================
Total params: 1,758,657
Trainable params: 1,755,841
Non-trainable params: 2,816
_________________________________________________________________
525/525 [==============================] - 350s 666ms/step - loss: 0.0237 - val_loss: 0.0133
```



(4) Visualise what the autoencoder has learned

In [67]:
```
#Visualize the reconstructions
def plot_image(image):
        plt.imshow(image,cmap='binary')
plt.axis('off')

def show_reconstructions(model,n_images=5):
    reconstructions = model.predict(valid_X[:n_images])
    fig = plt.figure(figsize=(n_images*1.5,3))
    for image_index in range(n_images):
        plt.subplot(2,n_images,1 + image_index)
        plot_image(valid_X[image_index].reshape(28,28))
        plt.subplot(2,n_images,1+ n_images + image_index)
        plot_image(reconstructions[image_index].reshape(28,28))

show_reconstructions(autoencoder)
```



MNIST Test

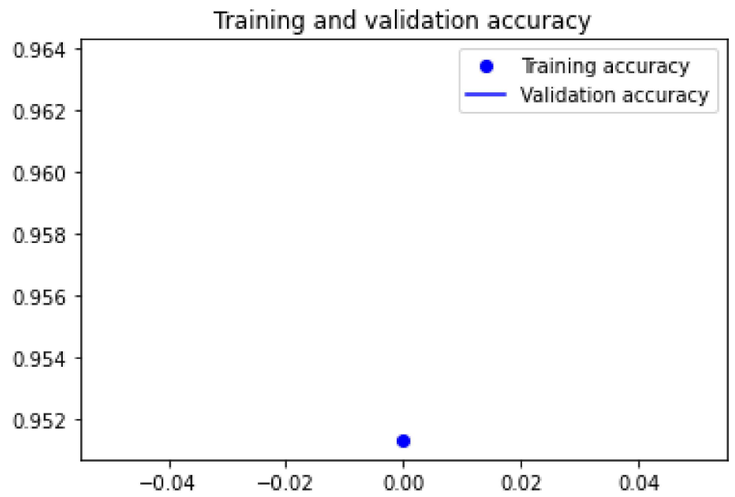
In [45]:
```
full_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(),metrics=['accuracy'])

print(full_model.summary())

#Train the model
classify_train = full_model.fit(train_X, train_label, batch_size=64,epochs=1,verbose=1,validation_data=(valid_X, valid_label))

#Save the model
full_model.save_weights('autoencoder_classification.h5')

#plot accuracy
accuracy = classify_train.history['accuracy']
val_accuracy = classify_train.history['val_accuracy']
loss = classify_train.history['loss']
val_loss = classify_train.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()

#Predictions
predicted_classes = full_model.predict(X_test)
predicted_classes = np.argmax(np.round(predicted_classes),axis=1)
```

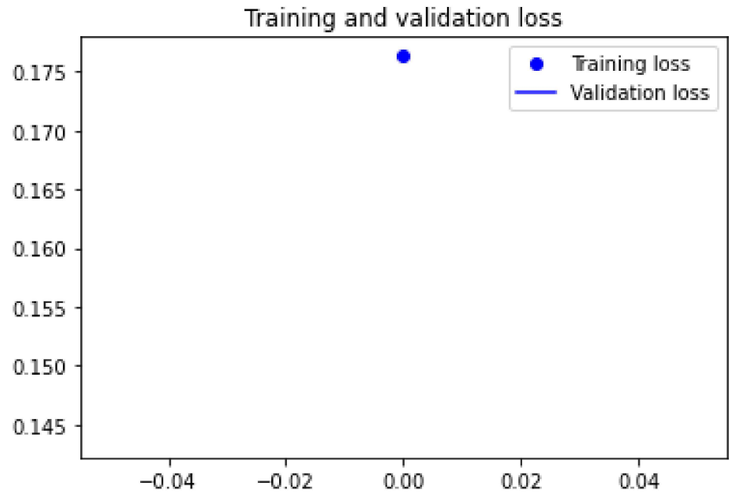
```
Model: "functional_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_79 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
batch_normalization_74 (Batc (None, 28, 28, 32)        128
_________________________________________________________________
conv2d_80 (Conv2D)           (None, 28, 28, 32)        9248
_________________________________________________________________
batch_normalization_75 (Batc (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_81 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
batch_normalization_76 (Batc (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_82 (Conv2D)           (None, 14, 14, 64)        36928
_________________________________________________________________
batch_normalization_77 (Batc (None, 14, 14, 64)        256
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_83 (Conv2D)           (None, 7, 7, 128)         73856
_________________________________________________________________
batch_normalization_78 (Batc (None, 7, 7, 128)         512
_________________________________________________________________
conv2d_84 (Conv2D)           (None, 7, 7, 128)         147584
_________________________________________________________________
batch_normalization_79 (Batc (None, 7, 7, 128)         512
_________________________________________________________________
conv2d_85 (Conv2D)           (None, 7, 7, 256)         295168
_________________________________________________________________
batch_normalization_80 (Batc (None, 7, 7, 256)         1024
_________________________________________________________________
conv2d_86 (Conv2D)           (None, 7, 7, 256)         590080
_________________________________________________________________
batch_normalization_81 (Batc (None, 7, 7, 256)         1024
_________________________________________________________________
flatten_4 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               1605760
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1290
=================================================================
Total params: 2,782,570
Trainable params: 2,780,650
```

Non-trainable params: 1,920

None

525/525 [==============================] - 240s 456ms/step - loss: 0.1763 - accuracy: 0.9513 - val_loss: 0.1438 - val_accuracy: 0.9637





```
       ImageId  Label
0            1      2
1            2      0
2            3      9
3            4      0
4            5      3
...        ...    ...
27995    27996      9
27996    27997      7
27997    27998      3
27998    27999      9
27999    28000      2

[28000 rows x 2 columns]
```

```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
<ipython-input-45-e8886bb2da73> in <module>
     35
     36 print(submission)
---> 37 submission.to_csv("submission.csv",index=False)

~\Anaconda3\lib\site-packages\pandas\core\generic.py in to_csv(self, path_or_buf, sep, na_rep, float_format, columns, header, index, index_label, mode, enc
oding, compression, quoting, quotechar, line_terminator, chunksize, date_format, doublequote, escapechar, decimal)
   3202             decimal=decimal,
   3203         )
-> 3204         formatter.save()
   3205
   3206         if path_or_buf is None:

~\Anaconda3\lib\site-packages\pandas\io\formats\csvs.py in save(self)
    182             close = False
    183         else:
--> 184             f, handles = get_handle(
    185                 self.path_or_buf,
    186                 self.mode,

~\Anaconda3\lib\site-packages\pandas\io\common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text)
    426         if encoding:
    427             # Encoding
--> 428             f = open(path_or_buf, mode, encoding=encoding, newline="")
    429         elif is_text:
    430             # No explicit encoding

PermissionError: [Errno 13] Permission denied: 'submission.csv'
```

Kaggle Score

In [68]:
```
results = pd.Series(predicted_classes,name="Label")
submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)

print(submission)
submission.to_csv("submission.csv",index=False)
```

```
       ImageId  Label
0            1      2
1            2      0
2            3      9
3            4      0
4            5      3
...        ...    ...
27995    27996      9
27996    27997      7
27997    27998      3
27998    27999      9
27999    28000      2

[28000 rows x 2 columns]
```

Username: sjdesai1 Score: 0.95585