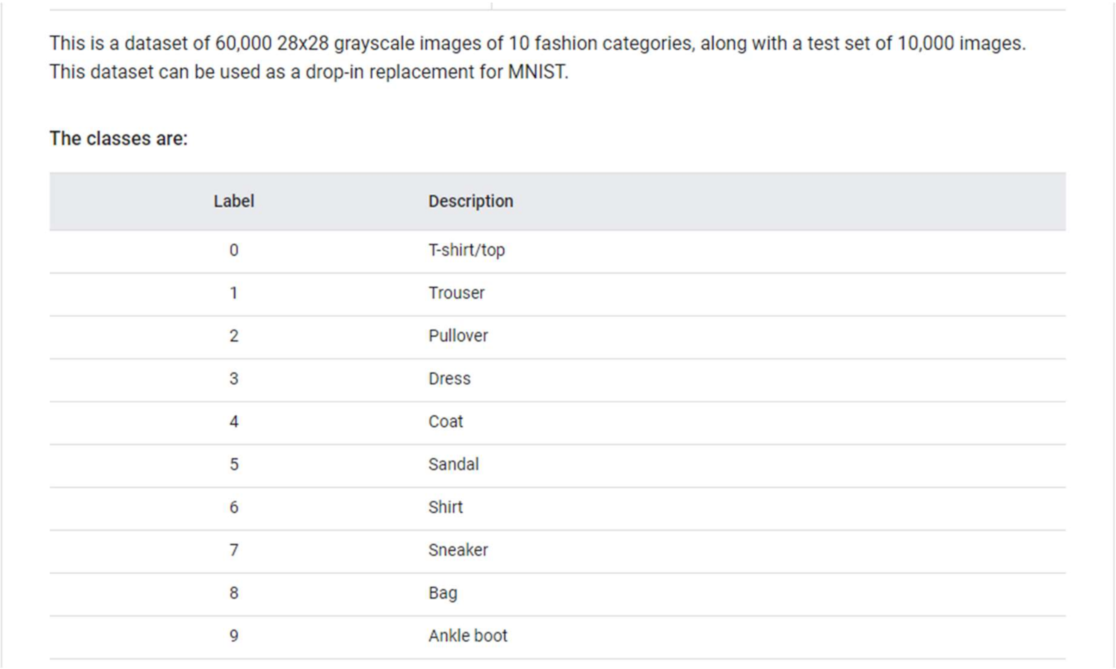
Applied Machine Learning
Assignment 2

Name	YAP LI XEN
Student ID	P7414389
Lecturer	MR. PAUL

PART B: DEEP LEARNING

(1) How is your prediction task defined? And what is the meaning of the output variable?

- The prediction task is to predict which category the image belongs to (to classify image)
- The output variables “Label” consists of 10 values (refer to screenshot below)
- This is multi-class variables



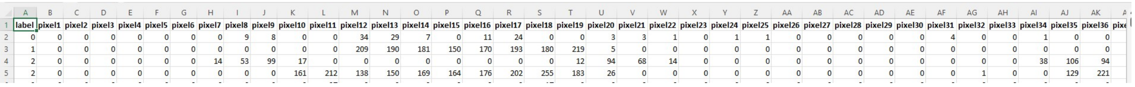
This is a dataset of 60,000 28x28 grayscale images of 10 fashion categories, along with a test set of 10,000 images. This dataset can be used as a drop-in replacement for MNIST.

The classes are:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

(2) How do you represent your data as features?

- Target: label
- Features: All columns except label. There are total of 784 columns (28 pixels * 28 pixels)

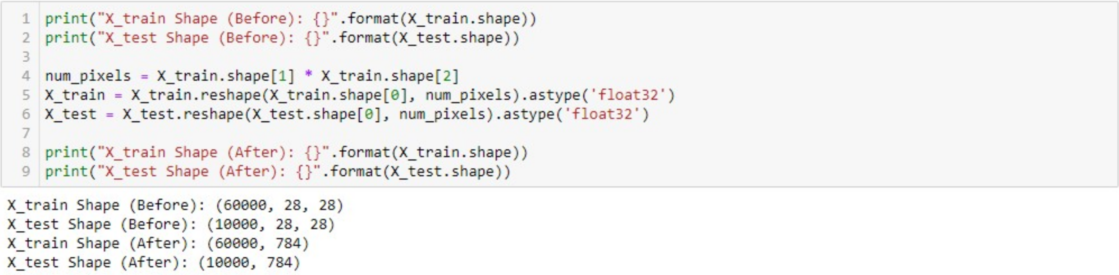


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	A
1	**label**	**pixel1**	**pixel2**	**pixel3**	**pixel4**	**pixel5**	**pixel6**	**pixel7**	**pixel8**	**pixel9**	**pixel10**	**pixel11**	**pixel12**	**pixel13**	**pixel14**	**pixel15**	**pixel16**	**pixel17**	**pixel18**	**pixel19**	**pixel20**	**pixel21**	**pixel22**	**pixel23**	**pixel24**	**pixel25**	**pixel26**	**pixel27**	**pixel28**	**pixel29**	**pixel30**	**pixel31**	**pixel32**	**pixel33**	**pixel34**	**pixel35**	**pixel36**	**pixe**
2	0	0	0	0	0	0	0	0	9	8	0	0	34	29	7	0	11	24	0	0	3	3	1	0	1	1	0	0	0	0	0	4	0	0	1	0	0	
3	1	0	0	0	0	0	0	0	0	0	0	0	209	190	181	150	170	193	180	219	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	2	0	0	0	0	0	0	14	53	99	17	0	0	0	0	0	0	0	0	12	94	68	14	0	0	0	0	0	0	0	0	0	0	0	38	106	94	
5	2	0	0	0	0	0	0	0	0	0	161	212	138	150	169	164	176	202	255	183	26	0	0	0	0	0	0	0	0	0	0	0	1	0	0	129	221	

(3) Did you process the features in any way?

- Reshaping

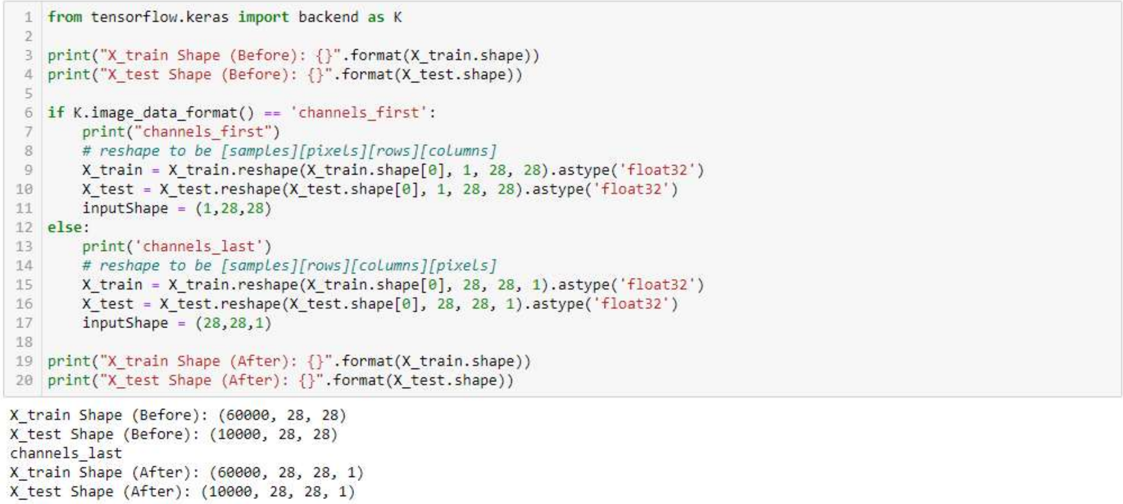
The original image pixel is 28*28, reshaping (Flatten) data to 784 vector for each image



```
print("X_train Shape (Before): {}".format(X_train.shape))
print("X_test Shape (Before): {}".format(X_test.shape))

num_pixels = X_train.shape[1] * X_train.shape[2]
X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')

print("X_train Shape (After): {}".format(X_train.shape))
print("X_test Shape (After): {}".format(X_test.shape))
```

```
X_train Shape (Before): (60000, 28, 28)
X_test Shape (Before): (10000, 28, 28)
X_train Shape (After): (60000, 784)
X_test Shape (After): (10000, 784)
```

This is to resolve different data shape (eg: channel first/last) issue.

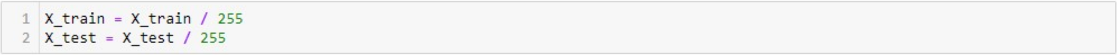


```
from tensorflow.keras import backend as K

print("X_train Shape (Before): {}".format(X_train.shape))
print("X_test Shape (Before): {}".format(X_test.shape))

if K.image_data_format() == 'channels_first':
    print("channels_first")
    # reshape to be [samples][pixels][rows][columns]
    X_train = X_train.reshape(X_train.shape[0], 1, 28, 28).astype('float32')
    X_test = X_test.reshape(X_test.shape[0], 1, 28, 28).astype('float32')
    inputShape = (1,28,28)
else:
    print('channels_last')
    # reshape to be [samples][rows][columns][pixels]
    X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
    X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
    inputShape = (28,28,1)

print("X_train Shape (After): {}".format(X_train.shape))
print("X_test Shape (After): {}".format(X_test.shape))
```

```
X_train Shape (Before): (60000, 28, 28)
X_test Shape (Before): (10000, 28, 28)
channels_last
X_train Shape (After): (60000, 28, 28, 1)
X_test Shape (After): (10000, 28, 28, 1)
```

- Scaling

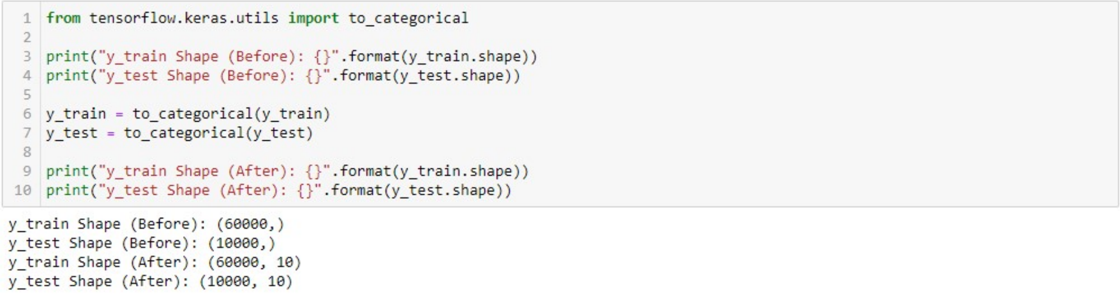
Normalize inputs from 0-255 to 0-1



```
X_train = X_train / 255
X_test = X_test / 255
```

- Encoding

Execute One Hot Encoding



```
from tensorflow.keras.utils import to_categorical

print("y_train Shape (Before): {}".format(y_train.shape))
print("y_test Shape (Before): {}".format(y_test.shape))

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

print("y_train Shape (After): {}".format(y_train.shape))
print("y_test Shape (After): {}".format(y_test.shape))
```

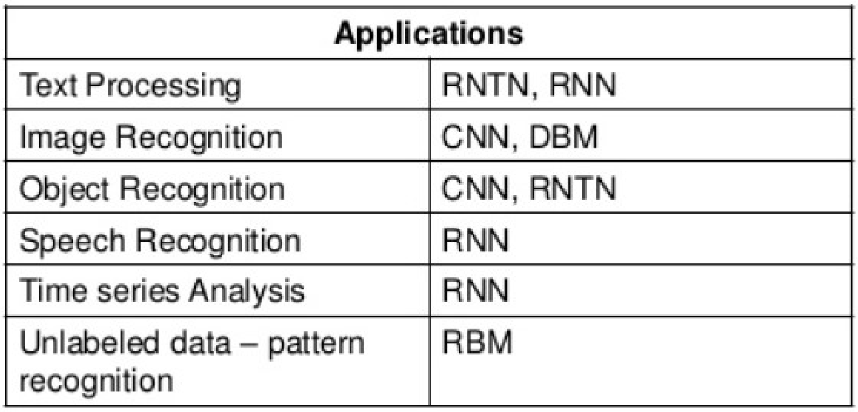
```
y_train Shape (Before): (60000,)
y_test Shape (Before): (10000,)
y_train Shape (After): (60000, 10)
y_test Shape (After): (10000, 10)
```

(4) Did you bring in any additional sources of data?

- No. Not necessary

(5) How did you select which DL model to use?

- Convolutional Neural Network (CNN) is primarily used for image processing and recognized by many in the industry.



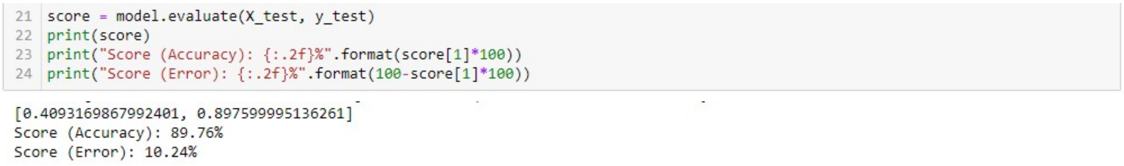
Applications	
Text Processing	RNTN, RNN
Image Recognition	CNN, DBM
Object Recognition	CNN, RNTN
Speech Recognition	RNN
Time series Analysis	RNN
Unlabeled data – pattern recognition	RBM

(6) Did you try to tune the hyperparameters of the learning algorithm, and in that case how?

- No

(7) How do you evaluate the quality of your system?

- Score the trained model using test data
- We can see the Accuracy Score and Error Score from the trained model while predicting the test data

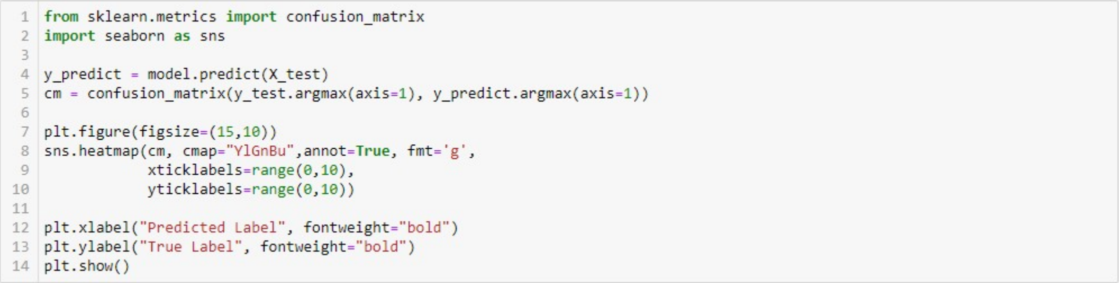


```
score = model.evaluate(X_test, y_test)
print(score)
print("Score (Accuracy): {:.2f}%".format(score[1]*100))
print("Score (Error): {:.2f}%".format(100-score[1]*100))
```

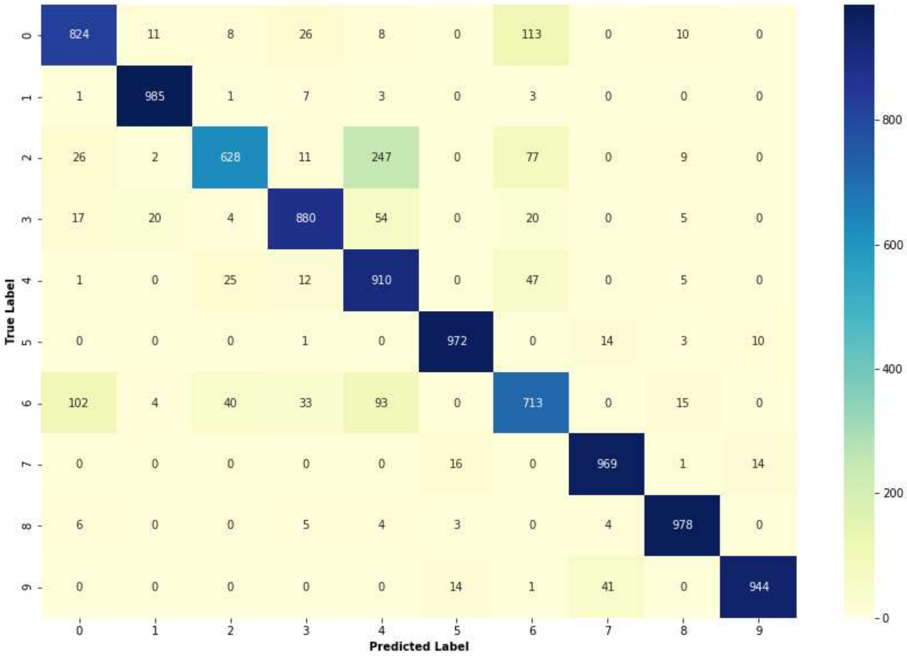
```
[0.4093169867992401, 0.897599995136261]
Score (Accuracy): 89.76%
Score (Error): 10.24%
```

(8) Can you say anything about the errors that the system makes? For a classification task, you may consider a confusion matrix.

- Using confusion matrix, it will show us the total number of
  - True Positive
  - False Positive
  - True Negative
  - False Negative



```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_predict = model.predict(X_test)
cm = confusion_matrix(y_test.argmax(axis=1), y_predict.argmax(axis=1))

plt.figure(figsize=(15,10))
sns.heatmap(cm, cmap="YlGnBu",annot=True, fmt='g',
            xticklabels=range(0,10),
            yticklabels=range(0,10))

plt.xlabel("Predicted Label", fontweight="bold")
plt.ylabel("True Label", fontweight="bold")
plt.show()
```



	0	1	2	3	4	5	6	7	8	9
0	824	11	8	26	8	0	113	0	10	0
1	1	985	1	7	3	0	3	0	0	0
2	26	2	628	11	247	0	77	0	9	0
3	17	20	4	880	54	0	20	0	5	0
4	1	0	25	12	910	0	47	0	5	0
5	0	0	0	1	0	972	0	14	3	10
6	102	4	40	33	93	0	713	0	15	0
7	0	0	0	0	0	16	0	969	1	14
8	6	0	0	5	4	3	0	4	978	0
9	0	0	0	0	0	14	1	41	0	944