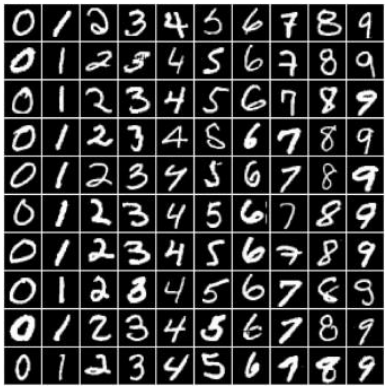
APVC – Desafio 3 (exercício semanal para avaliação)

Rede neuronal "*shallow*" para reconhecimento de dígitos manuscritos

Para este exercício pretende-se desenvolver uma rede neuronal clássica para classificar imagens de dígitos manuscritos. O *dataset* a utilizar é o MNIST[1], um dataset "fácil" muito utilizado que é também disponibilizado diretamente pelo tensorflow. Para realizar este exercício utilize como base o script `digitNet.py` fornecido. O script contém uma instrução para descarregar automaticamente o *dataset*, pelo que necessita de uma ligação à internet.



O *dataset* consiste em 60.000 amostras para treino e 10.000 amostras para teste. O dataset original não contém um conjunto de validação, mas deverá usar uma parte do conjunto de treino (e.g., 10.000 amostras) para validação. Cada amostra é uma imagem em tons de cinzento com uma dimensão de 28x28 pixels.

Embora os dados neste caso sejam imagens, pode-se aplicar uma rede neuronal clássica tratando o valor da luminância em cada pixel como se fosse um atributo. No entanto, e devido ao input serem imagens (e portanto matrizes bidimensionais), na definição da arquitetura da rede é necessária uma camada inicial extra, do tipo "`Flatten`", para transformar o input bidimensional num vetor que será posteriormente tratado pela rede neuronal da mesma maneira que nos casos vistos nas aulas. Para tal, na definição da arquitetura do modelo comece por colocar:

```
myModel = tf.keras.models.Sequential([
                layers.Flatten(input_shape=(28, 28)),
                …
```

Pretende-se que realize os seguintes passos:

1) Construir um modelo para uma rede neuronal capaz de classificar os dígitos representados nas imagens (10 classes, de 0 a 9, que correspondem a cada um dos dígitos);
2) Compilar a rede, escolhendo uma função de perda adequada e o algoritmo de otimização;
3) Treinar o modelo usando o conjunto de treino – espera-se que atinja uma taxa de acertos superior a 95%;
4) Mostrar a evolução da função de perda e dos acertos ao longo do treino da rede;
5) Calcular a taxa de acertos no conjunto de teste após treinado o modelo;
6) Mostrar a matriz de confusão e determinar quais dígitos que a rede confunde mais facilmente uns com os outros.

[1] http://yann.lecun.com/exdb/mnist/