# Final Project Deep Learning – Mark Peters

## Kunskapskontroll 1

## Questions:

1. **How are AI, Machine Learning, and Deep Learning related?**

A diagram of machine learning

Description automatically generatedAs shown in the visual from lecture 1, Machine Learning is a subset of AI, and Deep Learning is a subset of Machine Learning. AI is any machine that can mimic human or animal behaviour.

Machine Learning is a technique for machines to ‘learn’ from data. Deep Learning is a form of machine learning, but one that uses neural networks to identify patterns or predict the probabilities of certain outcomes.

1. **How are Keras and Tensorflow related?**

Tensorflow is a free and open-source software library created by Google to implement Machine Learning models and to solve complex numerical problems, whereas Keras is a deep learning API written in python for implementing and computing neural networks. Keras, therefore, is an interface for the Tensorflow library.

1. **What characterises regression problems?**

A regression problem is how to model one or more dependent variables (usually plotted on the ‘y’ axis), by means of a set of predictor variables on the X axis. The dependent variables have continuous values that can be plotted and predictions can be made from such equations as ‘root mean squared error’.

1. **What characterises classification problems?**

A classification problem is one where the goal is to assign a dependent variable to a class, either it is in a class (1, “spam2) or it is not (0, “not spam”). The output variable (also known as the target or label) is a discrete and categorical value, representing the class or category to which each data point belongs.

1. **What is a parameter? What is a hyperparameter?**

***Parameters*** are the variables or coefficients that a model uses to make predictions. ***Hyperparameters***govern the optimization process and the model’s behaviour. The first is *what* we learn, the latter *how* we learn.

1. **After deciding on the model selection and model evaluation, you use training, validation, and test datasets. Explain how the different parts can be used.**

Datasets need to split up into different training and test groups to avoid bias in your model; if the model is too sensitive to the data in the training set it will *overfit* new data and not reflect the true nature of the system the model is designed to explore.

***Training set***: This is the largest set of the three (about 70% of the total data), it needs this large as it is used to train your model and prepare it for application to unknown data.

***Validation set***: This set is not used during training but is instead used after every epoch in order to tune the hyperparameters to optimise performance.

***Test set***: This set is kept for the end of the training process, it is used to check the model’s final performance and generalisation to ‘unseen data’. It provides an estimate of how the model will respond to data in the real world.

1. **Explain the code:**



The ‘shape’ function gives you the dimensions of the X\_train dataset, the first value is the number of rows, the second value is the number of columns. By assigning the index ‘1’ to n\_cols we give it a value equal to the number of columns in the dataset.



‘Sequential’ is a function in the Keras API. We initialise it with this code, which means we are about to create a stack of layers used in the creation of a neural network (or deep learning) model.



Our first layer is our input layer. The Dense(100) means that there will be 100 neurons in this layer and that each will be connected to an input value. ReLU (Rectified Linear Unit) ‘relu’ is the activation function, it introduces non-linearity into the model, in this case it means that all negative values can transformed to ‘0’. The input shape is the number of input values (which will be the number of columns in the training dataset) for the first layer.



This is another regularisation technique (like lasso or ridge) that is there to avoid overfitting and so make the model more applicable to general, unseen data. This function, basically, turns off (sets to ‘0’) a certain number of cell outputs (in this case the probability of cell being set to zero is 0.2) between layers. This applies both to the forward and the back pass during training between the above and below layers.



This is another layer of cells; it has half the number of cells of the previous layer and is connected to each cell in that layer.



This is the output layer of this model. It has a value of ‘1’ which means that there is only a single output, and the activation function is ‘sigmoid’ which will give an output value of between 0 and 1. This indicates that this is a classification model designed to assign a variable to (or not to) a class.



When we are compiling the model, we are preparing it for fitting to our data. We choose an *optimiser* based on the nature of the problem (regression vs classification), the structure of our network, size of data, and a host of other factors. Its function is to fit the model quickly and accurately to the input data by affecting how the model’s weights are updated during training. We apply a ‘*loss function’* to the problem that we want to minimise, in this case we want to we want to predict to one of two classes. During training we will be able to see how the model is performing through the ‘*accuracy’* parameter.



With the ‘early stopping’ callback the model is told to quit once there is no improvement in results after a stated number of epochs, in this case the number is ‘5’. If the accuracy of the model has not improved after five consecutive epochs the model will stop running.



We then fit our training data to the model, putting aside 20% of the data for validation after each epoch. The maximum number of epochs (iterations over the entire training set) is 100, but if the accuracy of the model on the validation set doesn’t improve over 5 epochs the model will stop running.

A black and white text on a white background

Description automatically generated

1. **What is the purpose of ‘regularising’ a model?**

Regularising techniques are vital to a model to avoid overfitting and so make the model more applicable to new and unseen data. There are many forms (as I’ve discussed above) such as Dropout, Lasso, and ridge. The idea is to ‘penalise’ a model so that it doesn’t follow the training data too closely, and so avoid bias in the model. This can be done by setting values to zero, dropping values completely, or quitting the modelling before it over adjusts (in the case of ‘Early Stopping’).

1. **‘Dropout’ is a regularisation technique. What is it?**

This function, basically, turns off (sets to ‘0’) a certain number of cell outputs between layers. The value given to ‘dropout’ is a probability of any cell in a layer being set to zero. This applies both to the forward and the back pass during training between adjoining layers, though importantly, while it is not active in one pass it may be active in the next.

This prevents nodes from becoming over-specialised. Due to Dropout “the resulting neural network can be seen as an averaging ensemble of all these smaller neural networks” (Hands On Machine Learning, A Geron p366).

1. **‘Early Stopping’ is a regularisation technique. What is it?**

(As above). The ‘early stopping’ callback tells the model to quit once there is no improvement in results after a stated number of epochs. This stops the model from continuing to fit data after its best results have been achieved.

1. **Your colleague asks you what type of analysis is best for interpreting pictures. What do you say?**

The best type of network for identifying and recognising features in a picture is CNN or Convolutional Neural Networks.

1. **Briefly explain how a "Convolutional Neural Network" works.**

CNN’s copy, in a way, how humans identify images “by recognizing specific features or patterns anywhere in the image that distinguish each particular object class” (Introduction to Statistical Learning, Garth James et al, p407). For example, by recognising part of a face a being an eye, or a nose, or a mouth etc. The model identifies these ‘low-level’ features and then combines them to form higher-level features until we eventually create a probability of fitting the subject to a particular class or label.

1. **Your friend has a photo-album with 100 different pictures that contain, for example, tennis balls and zebras. How can they classify the pictures even though they don’t have any more pictures to train a model with?**

By using a pre-trained learning model, such as ResNet-50, it makes it possible to quickly classify pictures. ResNet-50 is a convolutional neural network (CNN) that has been trained on a huge dataset (ImageNet – consists of millions of images), in a much more comprehensive way than we can manage individually. This technique is called *transfer learning*.

Images need to be pre-processed so that they are the same shape and size of the ResNet pictures, as well as matching the expected pixel values. An extra dimension is added to the array of the uploaded image to match the input shape of the ResNet-50 model.

1. **Your colleague asks you what type of analysis is appropriate for carrying our sentiment analysis on text (such as film reviews). What do you say?**

For ’sentiment analysis’ I would advise them to use an RNN model (Recurrent Neural Network). In an RNN the input object is a *sequence*, in this case a sequence of words. Since we want a single analysis, e.g. is the review of a movie positive of negative, we output a vector based on the weight of the input words. Here “the order of the words, and closeness of certain words in a sentence, convey semantic meaning” (ISLP, p416). This is called a *sequence-to-vector* network, though in the case of a binary sentiment label this would be a scaler value (good or bad).

1. **Briefly explain how a Recurrent Neural Network works.**

The first neural networks we learned were feedforward networks, where the activations flow in only one direction, from the input layers to the output layers. RNN’s work in a similar fashion, but they also have connections pointing backwards (which is where the ‘recurrent’ comes from). Each recurrent neuron (cell) has two sets of weights; one from the input from a cell in an underlying layer, and one from the outputs of a previous time step, which is why the cells can be said to have a *memory*. From this we can create the different model formats: sequence-to-sequence, sequence-to-vector, vector-to-sequence, and encoder-decoder.

This memory makes it a powerful tool in predicting time series data, in sentiment analysis, transcription from recorded speech, language translation, and identifying words and figures from handwritten notes.

1. **Explain the following code:**



Neural networks are models that take a lot of time to train, and this training is its currency as the more data it is trained with, the stronger and more accurate it becomes. It is vital then, that we can save the model so it can be used and re-used when required.

It may also be that your model is constantly developing (think of the ResNet-50 model that is constantly being updated with new images from imagenet) so saving the learned parameters is key. Keras makes is possible to save a model as a HDF5 file where its architecture (cell numbers, layers, configurations) can be saved, as well as its weights (tuned hyperparameters).



That saved model can then be loaded when required and either used to make predictions on new data or continue to train with new instances of data from an imported dataset.