Practical 9

**Deep Learning in Weka**

# What are we doing?

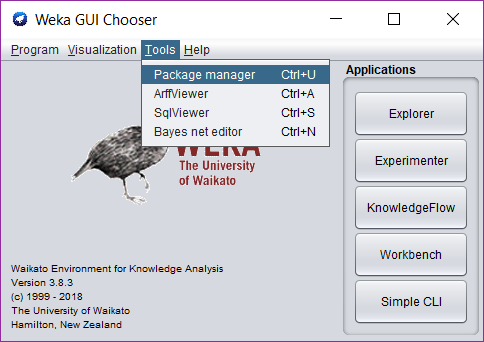
Trying out some simple neural networks to recognise handwritten digits using Weka.

**Submission:**

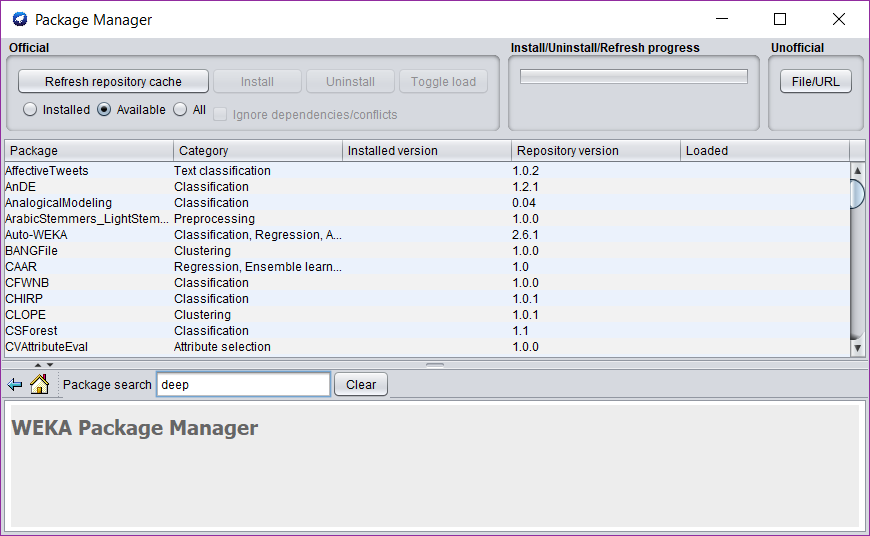
Please submit a single Word doc with your solutions to the 6 questions.

# Install the deeplearning4j package

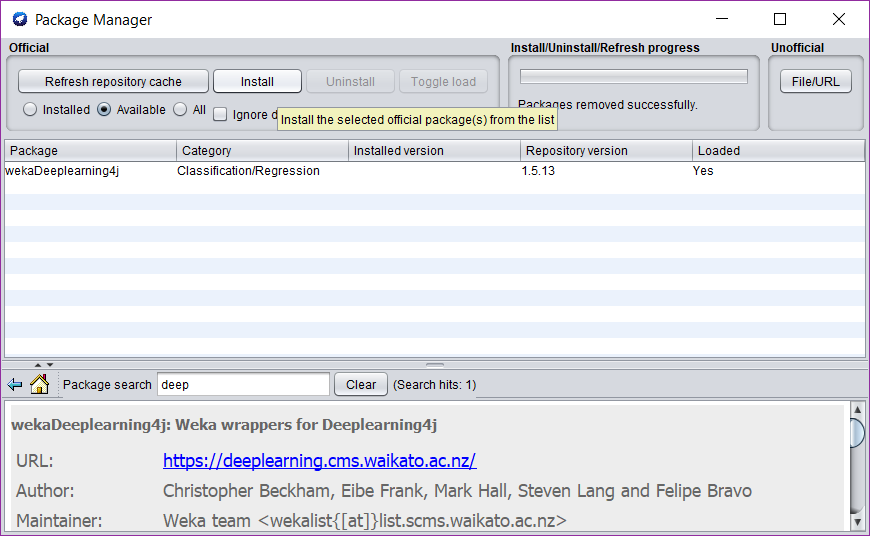
We can do deep learning in Weka using the deeplearning4j package. Open Weka and go to Tools🡪Package manager



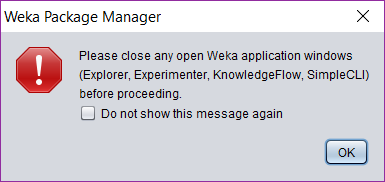
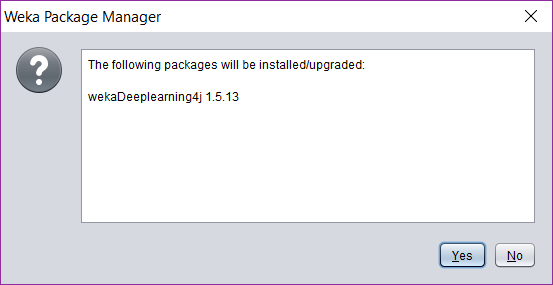
In the “Package search” box, type “deep” and hit enter.

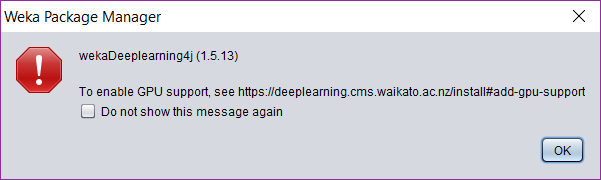


You should see “wekaDeeplearning4j” in the package list. Click the Install button.



You’ll get a few confirmation boxes. Just click yes/OK through all these.



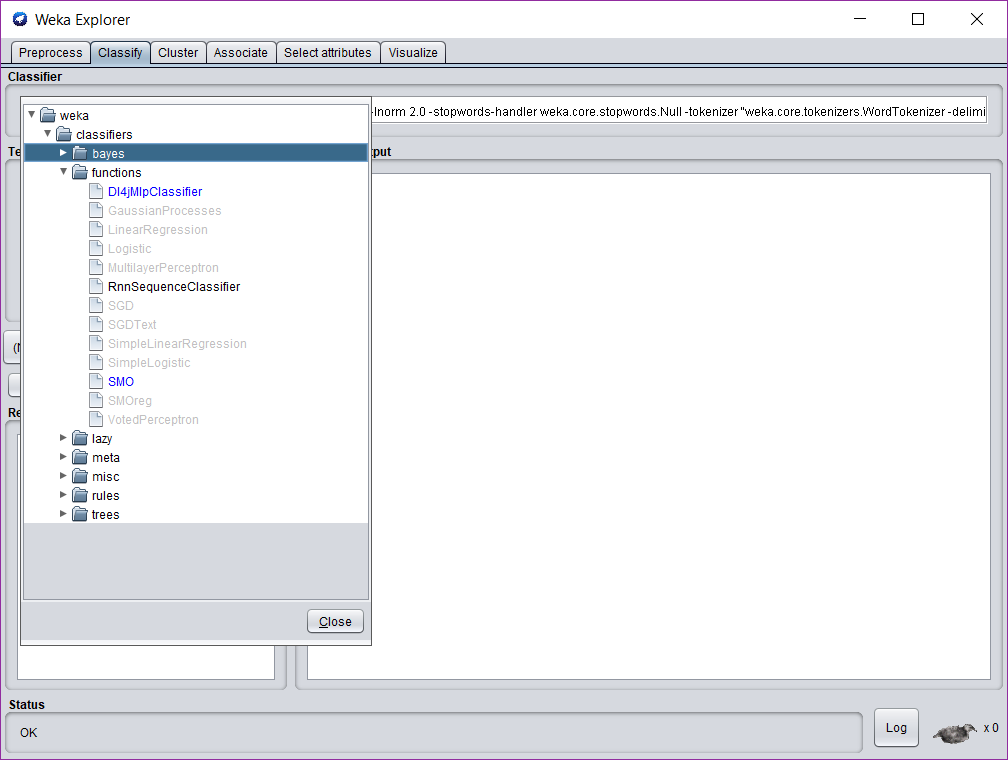


# Try neural network models on MNIST handwriting images

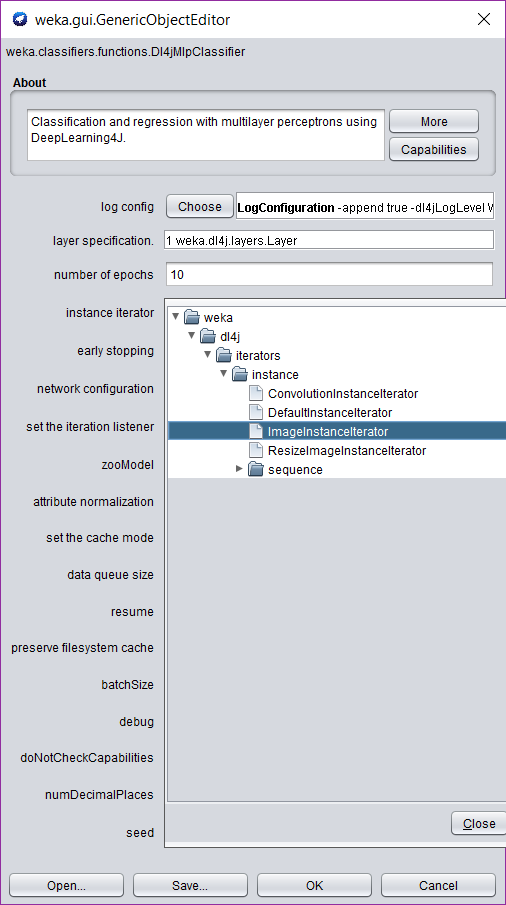
Download “mnist-minimal.zip” and unzip. You’ll see an ARFF file **mnist-3k.arff**. This is a “meta” ARFF file – instead of containing data, it contains filenames and labels. You should also find a folder called mnist-3k which contains 3000 PNG files of handwritten digits.

Classification steps

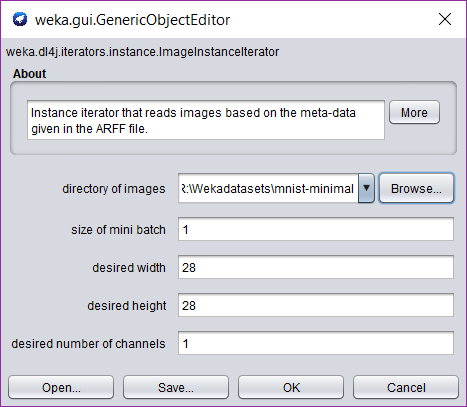
1. Open the ARFF file in Weka Explorer and go to Classify. Choose weka🡪classifiers🡪functions🡪Dl4jMlpClassifier to access the neural network classifier.



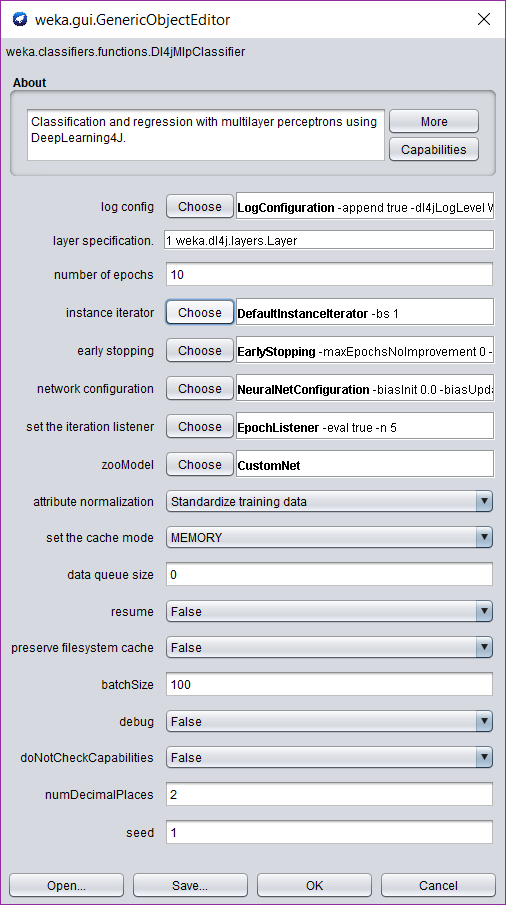
1. We have to get the classifier to classify based on the images, rather on the filename strings themselves. To do this, open the properties for the classifier and click “Choose” next to **instance iterator** and click on **ImageInstanceIterator**.



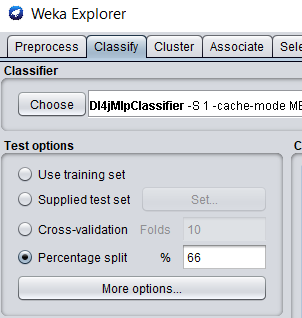
1. We now need to tell the ImageInstanceIterator which folder to look in. Left-click on the text “ImageInstanceIterator -height 28…” to open the ImageInstanceIterator properties editor. Click “Browse…” next to **directory of images** and find the folder **mnist-3k** with the JPG files.



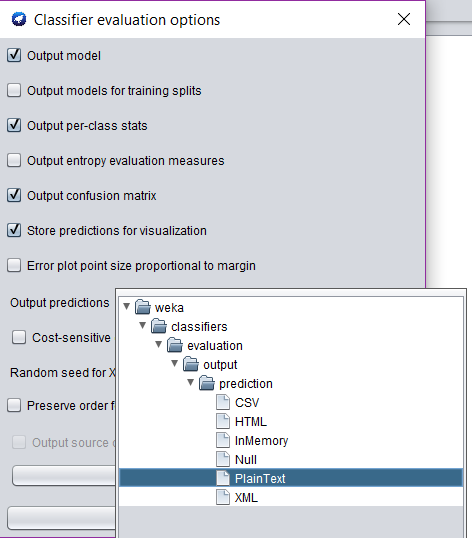
1. Check that your properties editor looks like the following, then click “OK” to exit.

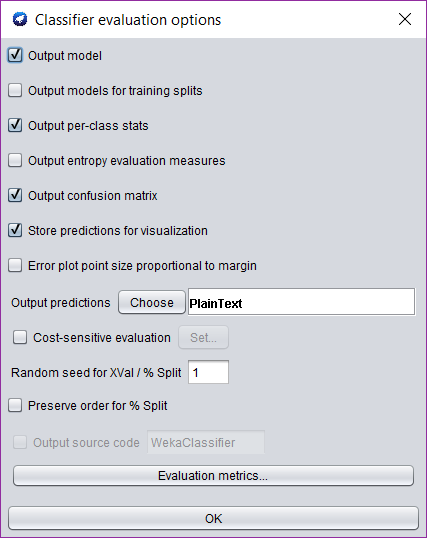


1. Choose “Percentage split” for the test options and click “More options”.

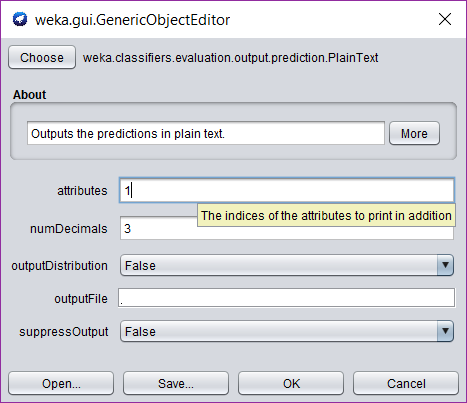


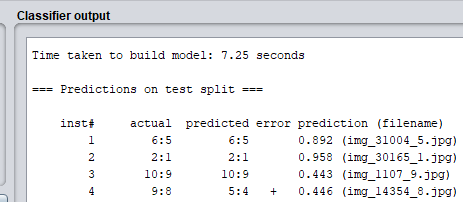
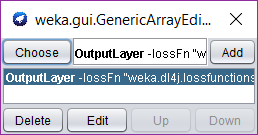
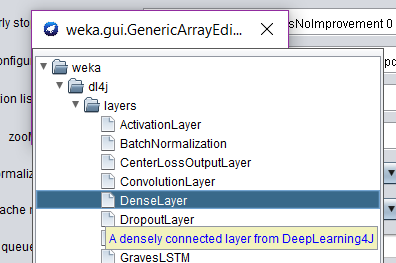
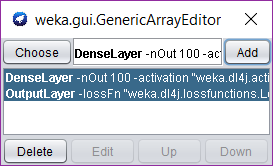
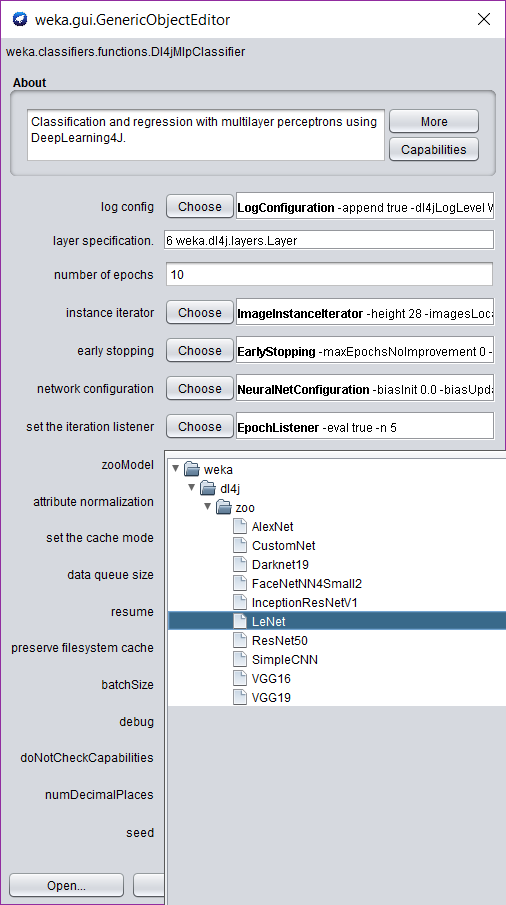
1. Click “Choose” next to **Output predictions** and then click **PlainText**



1. Now click where it says PlainText next to OutputPredictions to open the output properties. We want to output the filenames of the tested digits so we can inspect which ones the classifier fails on. Enter **1** next to the attributes. The filename is attribute 1, hence this will give us filename output.  
   

CLICK HERE



1. Click OK through the boxes to get back to the main Weka Explorer window.
2. Click “Start” to do the first classification.   
   **Question 1: What is the accuracy of this initial model?**
3. Scroll up in the classifier output to find the predictions made on the test data. You should see something like the following:  
   
4. This output shows all the predictions. The fourth column shows a “+” when the classifier has made an error, for example, instance #4 above, where the classifier saw an image of an **8** but classified it as a **4**. The fifth column gives a number indicating how confident the classifier is – in this case it’s 0.446 or 44.6%. Finally, in parentheses the output shows the filename.
5. Look through your output and find three cases where the classifier got it wrong. Search in the mnist-minimal folder for the filename and examine the image.  
   **Question 2: Copy three images that the classifier got wrong to your solution doc. Do these examples look like they would be difficult to classify (e.g. do they like someone might mistake them for a different number?)  
     
   Question 3: How long did it take to build the model?**
6. The model we used above is very shallow – it only includes an input and output layer. Let’s create a new model now with a hidden layer. Go back to the classifier properties and click the text next to “layer specification”. This will open the layer specification editor.  
   
7. Click “Choose” and then click “DenseLayer”.  
   
8. Now click on the “DenseLayer -nOut 0…” text. Set **number of outputs** to 100 and click “OK”.
9. Now click the “Add” button. You should see the following:  
   
10. Click the **X** in the corner to close this window and go back to the classifier properties editor, where you should see that **layer specification** now reads **2 weka.dl4j.layers.Layer**. Click “OK” to go back to the main Weka Explorer window.
11. Click “Start” to run this model.
12. **Question 5: Compared to the first classifier, was this more accurate, and how long did it take to train?**
13. For our third and final model, go back to the classifier properties editor, and “Choose” next to **zooModel**. Next click **LeNet** which is a simple convolutional neural network architecture. Click “OK” to get back to the main Weka Explorer window, and then click “Start” to train this network. **Note: this one may take up to 30 minutes.** (If you think this is a long time, neural networks deployed in industry can take weeks to train.)  
    
14. **Question 6: Pick some example images that the LeNet classifier got wrong. Do these look “trickier” to classify than the images that the original classifier failed on? What is the overall accuracy of this classifier?**