ACS61011 Deep Learning Project

General Assignment Information

Assignment weighting

20%

Assignment summary

The assignment is to design, implement and evaluate a deep learning system.

Assignment start date

Tuesday 14th March (Week 6)

Assignment due date

The assignment is due at 23:59 pm on Monday 27th March (start of week 8).

Assignment supporting materials and data

All assignment instructions, supporting material and data are on Blackboard in the ACS61011 course pages, under the Project section

Submission

You will have to submit a short Technical Report as an MSWord or PDF document in Blackboard under the Turnitin link in the ACS61011 Project section by the due date.

Penalties for Late Submission

Late submissions will incur the usual penalties of a 5% reduction in the mark for every working day (or part thereof) that the assignment is late and a mark of zero for submission more than 5 working days late.

Unfair Means

The assignment should be completed individually. You should not discuss the assignment with other students and should not work together in completing the assignment. The assignment must be wholly your own work. Any suspicions of the use of unfair means will be investigated and may lead to penalties. See http://www.shef.ac.uk/ssid/exams/plagiarism for more information.

Special Circumstances

If you have medical or personal circumstances which cause you to be unable to submit this assignment on time or that may have affected your performance, follow the guidance at https://www.sheffield.ac.uk/ssid/forms/circs

Help

We will use the lab sessions in Weeks 6 and 7 for drop-in support in the usual lab room.

If you need further clarifications on the assignment then please discuss the issue with me after a lecture or email me s.anderson@sheffield.ac.uk or make an appointment to meet.

Specific assignment information and instructions

The aim of this project is to develop a deep learning system to perform automated speech recognition. This project can be done in either Matlab or Python. If you would rather use different data that might be possible – speak to me first though to check s.anderson@sheffield.ac.uk

Data

Input data: Raw speech has been pre-processed into spectrograms of speech 'images' (power vs time vs frequency). This is the input data (Figure 1). The data is split into training and validation sets (Figure 2).

Output data: There are 12 word categories (i.e. 12 classes). This is the output data.

Files are provided on Blackboard in the Project section in the file speechImageData.zip.

- Matlab users: Unzip speechImageData.zip and put it into your working directory for the project.
- Python users: there is a link to a Colab file on Blackboard, which pulls in the data from github.

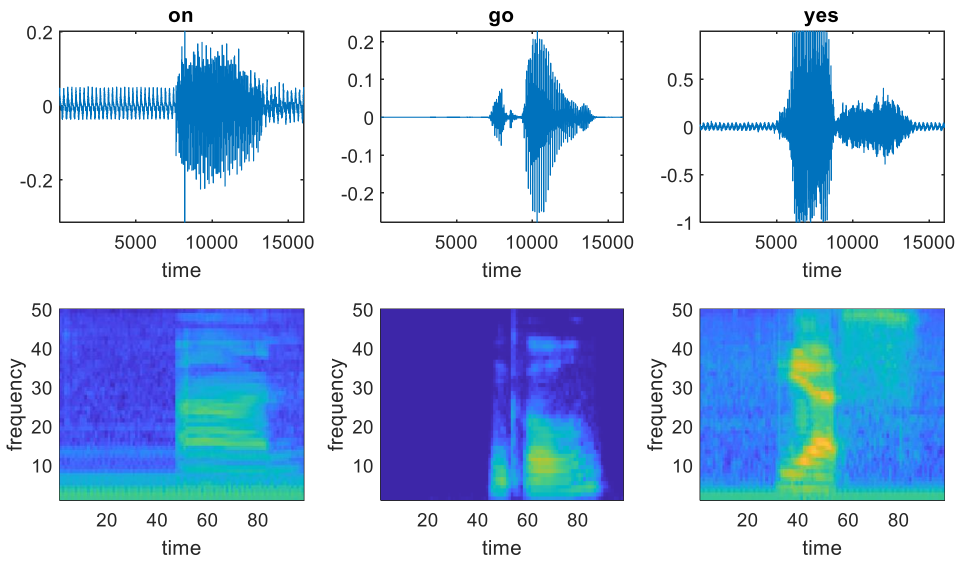


Figure 1. Examples of speech waveforms in the time-domain (top) and corresponding speech image spectrogram that forms the input data (bottom). The spoken word is given in the title and forms the class label output: 'on', 'go' and 'yes'.

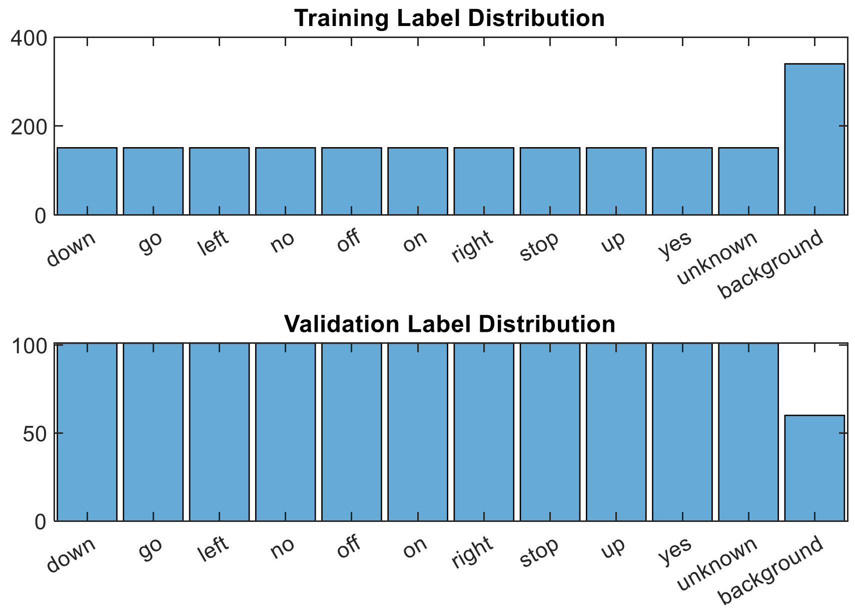


Figure 2. The initial training and validation data has the distribution shown in the plots, i.e. 12 words (therefore 12 classes) and nearly 200 training data pairs per word and 100 validation data pairs per word.

Initial Code and Matlab versus Python Tools

Initial Matlab and Python code to import the data is provided on Blackboard.

Key tip: here is a tip based on 'expert' knowledge. Words can start and stop at different times in the data recording – therefore it is well-known in this domain that it makes a huge difference if there is a maxpool layer as the final layer of feature extraction, before the fully connected -> softmax layers. This maxpool layer should pool features across the time-dimension – for this dataset it should look like this:

```
timePoolSize=12
```

Matlab: `maxPooling2dLayer([timePoolSize,1])`

Python:

```
model.add(MaxPooling2D(pool_size =(timePoolSize, 1), strides=(1, 1), padding
= 'same'))
```

The variable timePoolSize=12 is empirically tuned here to give reasonable performance.

Useful background reading

Here is a source that provides useful background reading:

- The use of convolutional neural networks for speech recognition is described in Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22(10), 1533-1545.

Technical report

Words: For each distinct task or subtask specified in the mark scheme (over the page) you should write just enough to explain what you have done and justify your design choices – a bullet point list is fine.

Figures: You should include a variety of figures for each task to evidence what you have done – e.g. 1-4 per task should probably be sufficient.

- To evidence your network design you can visualise the network with analyzeNetwork in Matlab.
- To demonstrate training/validation performance you should use a screenshot of the training/validation plot that the Deep Learning toolbox produces during training.
- To demonstrate performance you should also include a confusion plot for each task.
- Any other figures as necessary
- Example figures are given at the end of this report.

Code: You should include all your code in the report document – copy and paste the code into the report document and clearly label it. This helps for both marking and plagiarism detection in Turnitin.

Tasks and Mark Scheme

The specific tasks and corresponding mark scheme are given in the table below.

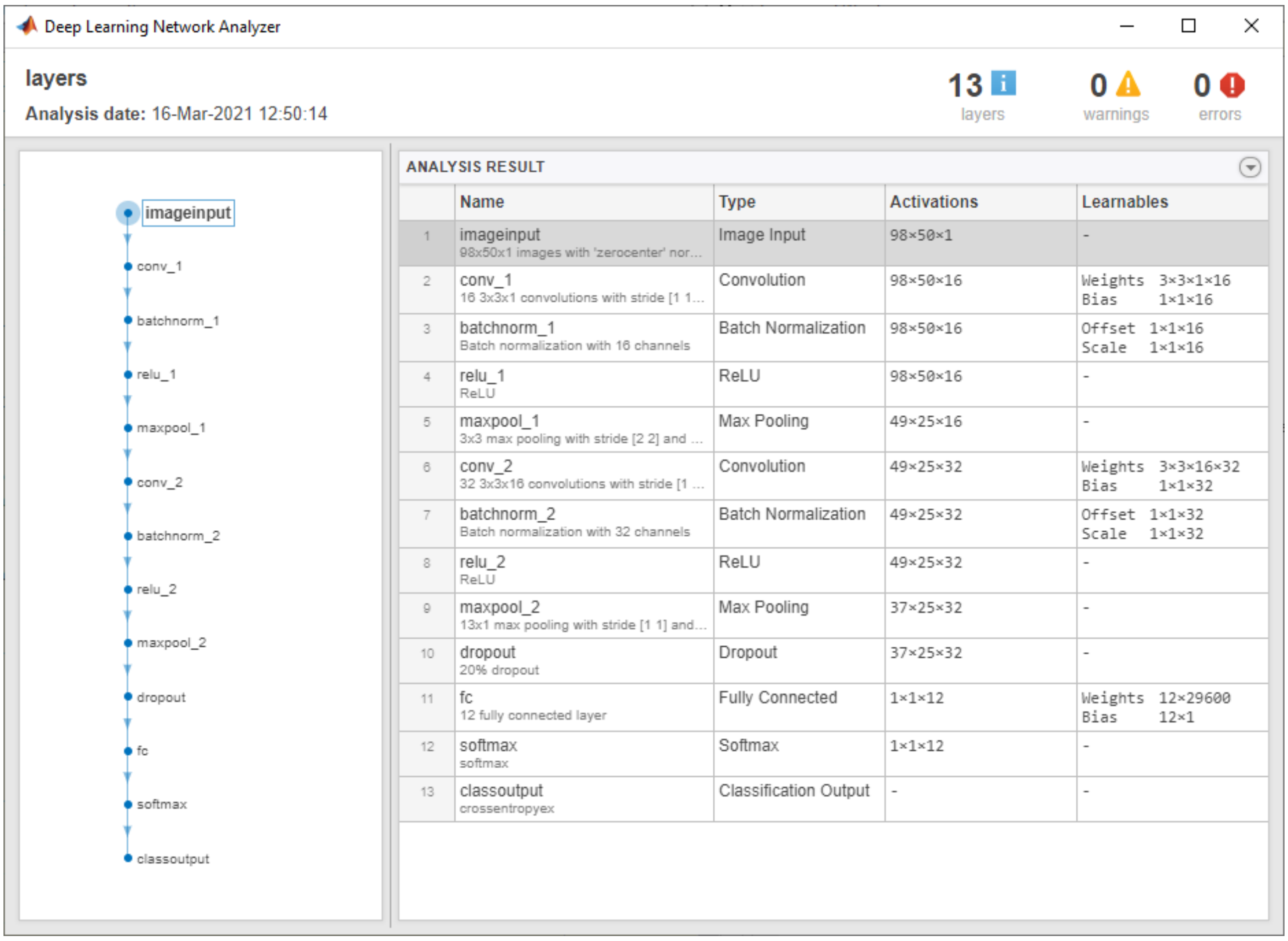
It is up to you to choose what amount of work you do. For each task, the mark within a grade boundary will be moderated based on your results and code.

Note, I reserve the right to mark at a lower level if the tasks are done very poorly (or e.g. presented poorly).

Level	Mark Range	Task/Assessment Description
1	0-49%	An attempt at the project to design, implement and evaluate a basic deep learning classifier system, to automatically recognise spoken words from spectrogram image inputs, which achieves an accuracy of <60% on the validation data. Little or no results/code/evidence of model accuracy.
2	50%	Design, implement and evaluate a basic deep learning classifier system to recognise spoken words from spectrogram image inputs and achieve an accuracy of >=60% on the validation data. This is a basic baseline model that should be quick to develop and should ensure you can familiarise with the problem and data.
		Only go to the next level once you have completed the preceding task
3	50-60%	Achieve level 2 plus the following task: Use a grid search <u>or</u> random search (not both – unless you want to!) to perform hyperparameter tuning in your model of: -the number of layers and -number of filters per layer You can just use a 2x2 or 3x3 grid search here, or equivalently random search 4 to 9 different networks, just to evidence that you can do it, because anything more and you might run into computational difficulties because these models can take on the order of 5-10 minutes each to train. Once completed, take your best model forward to the next stage.
		Only go to the next level once you have completed the preceding task
4	60-70%	Achieve level 3 plus the following tasks: -Reduce the computational complexity of your level 3 model using methods such as depthwise separable convolutions and 1x1 convolutions. Evidence that you have reduced the model complexity (in terms of number of parameters) compared to the model from Level 3. -Design, implement and evaluate a model averaging scheme to regularise your model and improve generalisation.
		Only go to the next level once you have completed the preceding tasks
5	70-100%	Do level 4 and then do **an open-ended extension of your choice**, e.g.: (Note: you do not have to do any/all of the following – they are just ideas) -Try and improve the model using more advanced model designs such as Inception modules or skip connections from ResNet. (But do not just retrain an established model with transfer learning, which is very simple to do – design your own model). A systematic investigation into the benefits of the design will score higher. -Perform a more sophisticated hyperparameter search using e.g. Bayesian optimization. -Perform multi-objective hyperparameter optimisation maximising accuracy and minimising model complexity. Note that it is not worth that much more to repetitively do tasks of a similar difficulty level so do not spend a long time doing lots of extensions of a similar level of difficulty. -You can check your idea with me if unsure s.anderson@sheffield.ac.uk

Example figures for the technical report

Analyzenetwork figure



layers

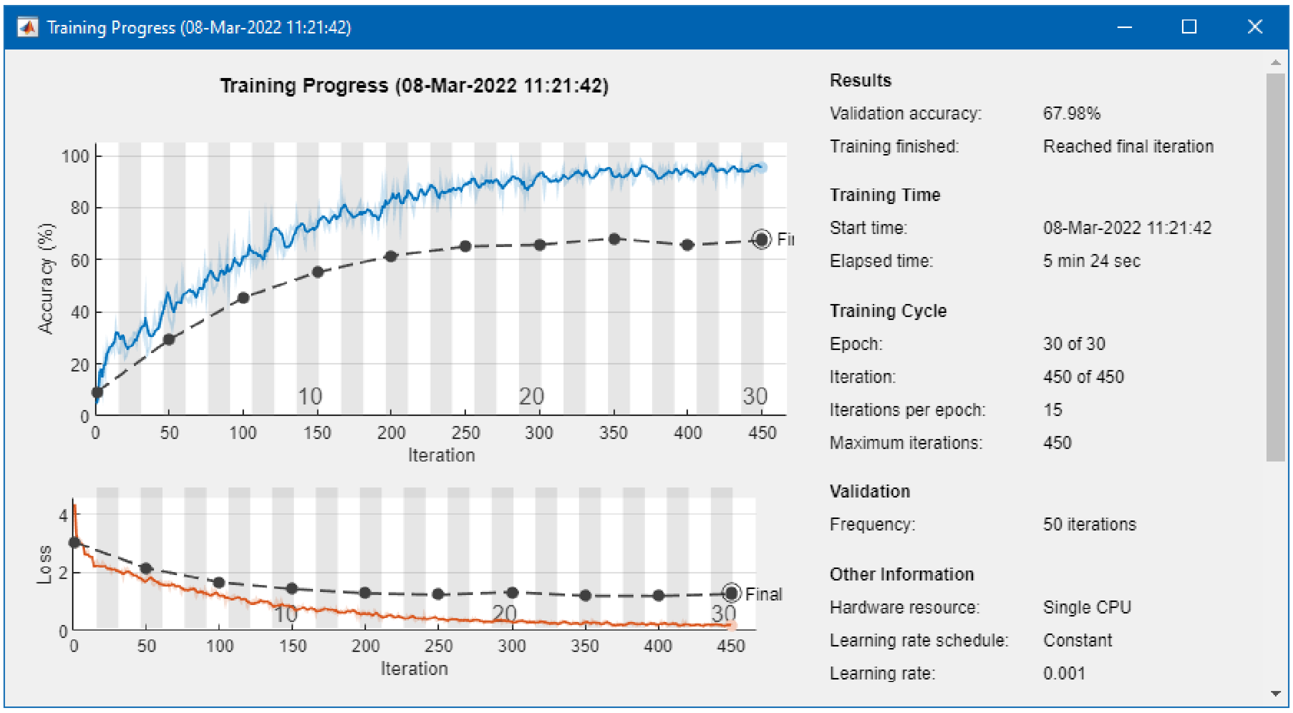
Analysis date: 16-Mar-2021 12:50:14

13 layers | 0 warnings | 0 errors

ANALYSIS RESULT

	Name	Type	Activations	Learnables
1	imageinput 98x50x1 images with 'zerocenter' nor...	Image Input	98×50×1	-
2	conv_1 16 3x3x1 convolutions with stride [1 1...	Convolution	98×50×16	Weights 3×3×1×16 Bias 1×1×16
3	batchnorm_1 Batch normalization with 16 channels	Batch Normalization	98×50×16	Offset 1×1×16 Scale 1×1×16
4	relu_1 ReLU	ReLU	98×50×16	-
5	maxpool_1 3x3 max pooling with stride [2 2] and ...	Max Pooling	49×25×16	-
6	conv_2 32 3x3x16 convolutions with stride [1 ...	Convolution	49×25×32	Weights 3×3×16×32 Bias 1×1×32
7	batchnorm_2 Batch normalization with 32 channels	Batch Normalization	49×25×32	Offset 1×1×32 Scale 1×1×32
8	relu_2 ReLU	ReLU	49×25×32	-
9	maxpool_2 13x1 max pooling with stride [1 1] and...	Max Pooling	37×25×32	-
10	dropout 20% dropout	Dropout	37×25×32	-
11	fc 12 fully connected layer	Fully Connected	1×1×12	Weights 12×29600 Bias 12×1
12	softmax softmax	Softmax	1×1×12	-
13	classoutput crossentropyex	Classification Output	-	-

Training/validation plot



Results

Validation accuracy:	67.98%
Training finished:	Reached final iteration

Training Time

Start time:	08-Mar-2022 11:21:42
Elapsed time:	5 min 24 sec

Training Cycle

Epoch:	30 of 30
Iteration:	450 of 450
Iterations per epoch:	15
Maximum iterations:	450

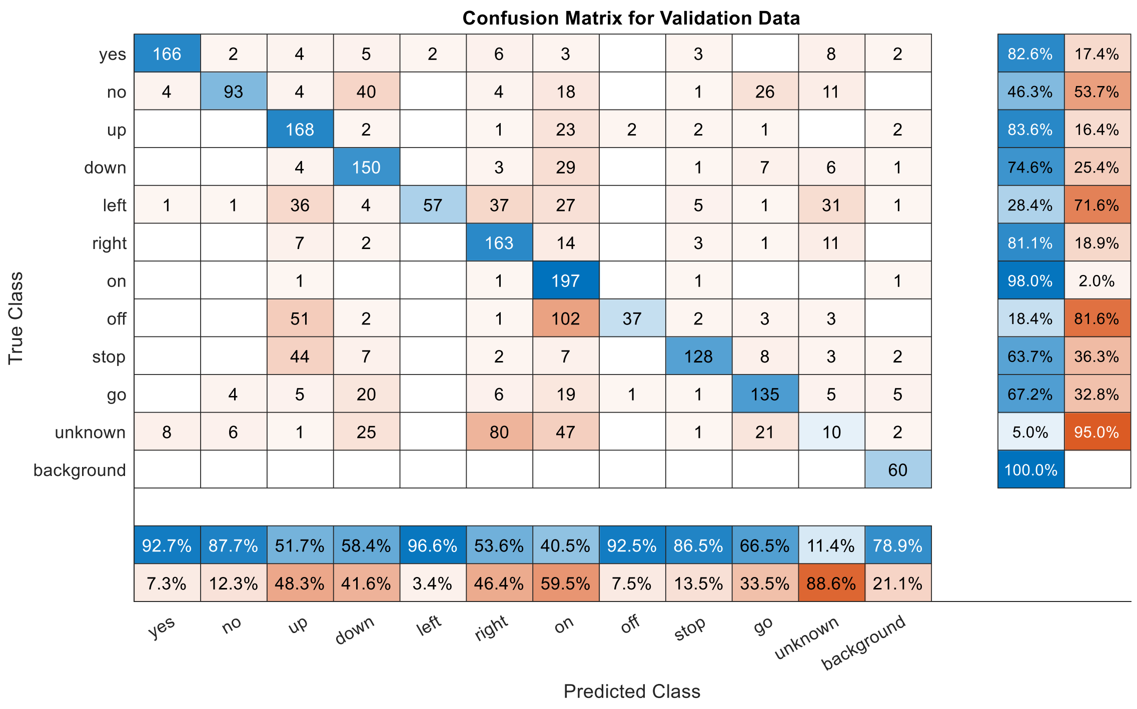
Validation

Frequency:	50 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001

Confusion plot



	yes	no	up	down	left	right	on	off	stop	go	unknown	background		
yes	166	2	4	5	2	6	3		3		8	2	82.6%	17.4%
no	4	93	4	40		4	18		1	26	11		46.3%	53.7%
up			168	2		1	23	2	2	1		2	83.6%	16.4%
down			4	150		3	29		1	7	6	1	74.6%	25.4%
left	1	1	36	4	57	37	27		5	1	31	1	28.4%	71.6%
right			7	2		163	14		3	1	11		81.1%	18.9%
on			1			1	197		1			1	98.0%	2.0%
off			51	2		1	102	37	2	3	3		18.4%	81.6%
stop			44	7		2	7		128	8	3	2	63.7%	36.3%
go		4	5	20		6	19	1	1	135	5	5	67.2%	32.8%
unknown	8	6	1	25		80	47		1	21	10	2	5.0%	95.0%
background												60	100.0%	
	92.7%	87.7%	51.7%	58.4%	96.6%	53.6%	40.5%	92.5%	86.5%	66.5%	11.4%	78.9%		
	7.3%	12.3%	48.3%	41.6%	3.4%	46.4%	59.5%	7.5%	13.5%	33.5%	88.6%	21.1%		