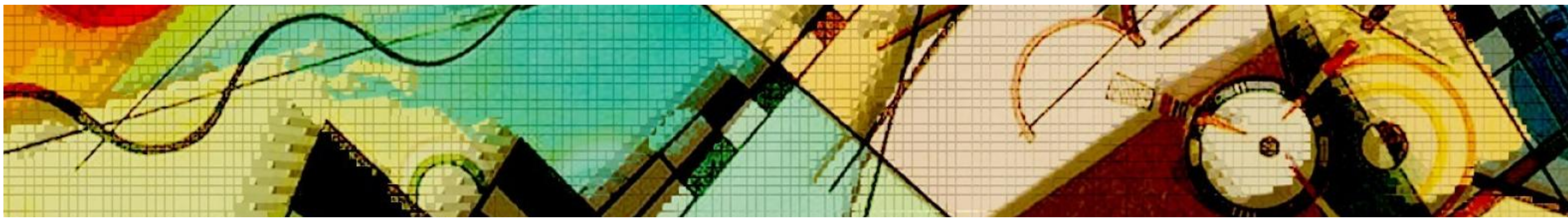
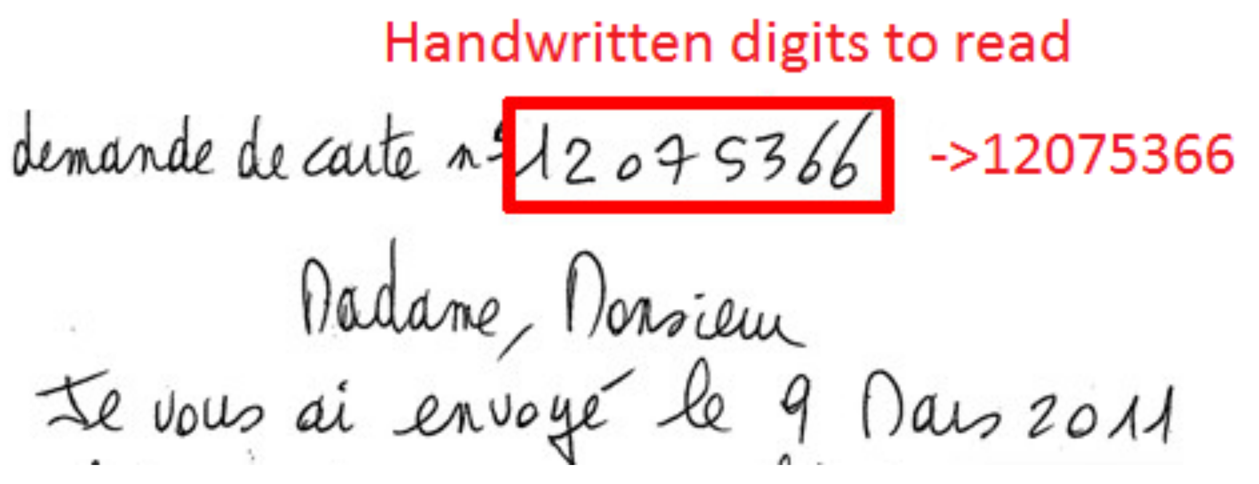


CSC411 Project 2: Deep Neural Networks for Handwritten Digit and Face Recognition

For this project, you will build several neural networks of varying depth, and use them for handwritten digit recognition and face recognition. You may `import` things like `numpy` and `matplotlib`, but the idea is to implement things "from scratch": you may not import libraries that will do your work for you, unless otherwise specified


Handwritten digits to read
demande de carte n° 12075366
->12075366
Madame, Monsieur
Je vous ai envoyé le 9 Mars 2011

The input: digits

You will work with the MNIST dataset. The dataset is available in easy-to-read-in format here, with the digits already seprated into a training and a test set. I recommend you divide the data by 255.0 (note: the .0 is important) so that it's in the range 0..1.

Handout code: digits

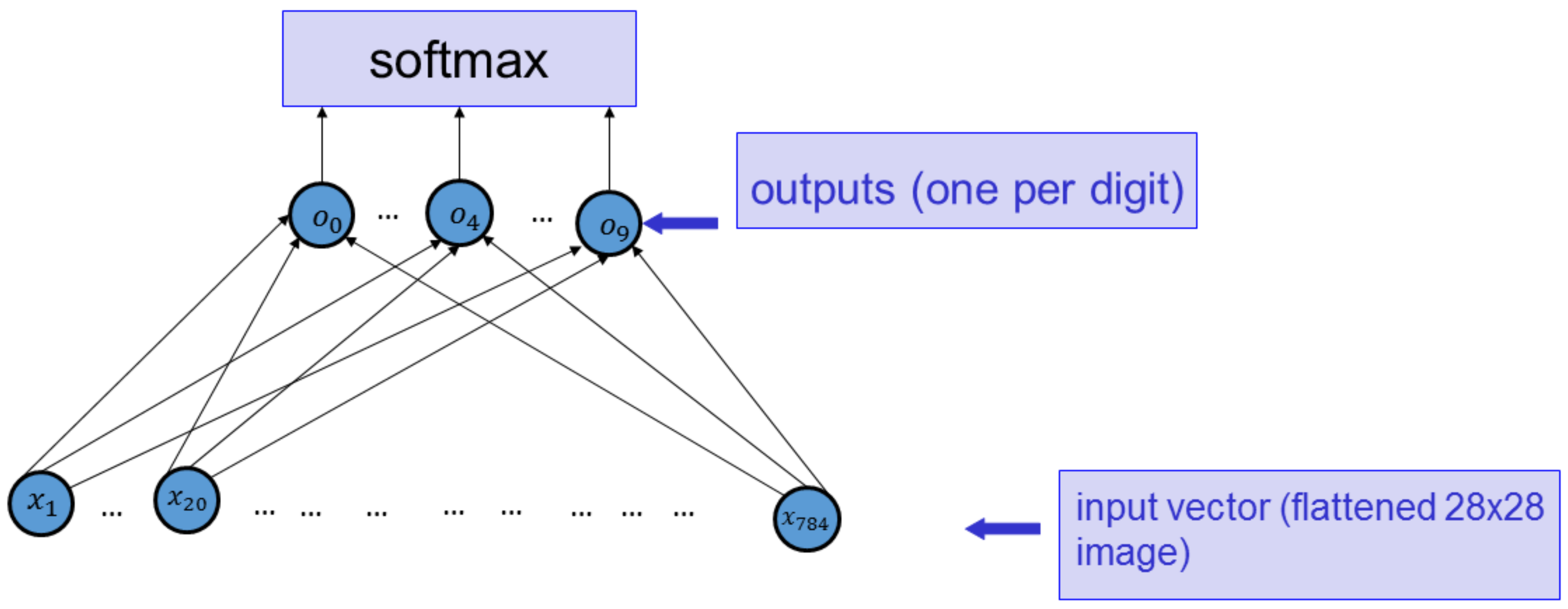
I am providing some code here (data: snapshot50.pkl. You need to read it using `pickle.load(open("snapshot50.pkl", "rb"), encoding="latin1")` in Python 3). The code provides a partial implementation of a single-hidden-layer neural network with tanh activation functions in the hidden layer that classifies the digits. **The code is not meant to be run as is**. I am simply providing some functions and code snippets to make your life easier.

Part 1 (2 pts)

Describe the dataset of digits. In your report, include 10 images of each of the digits. You may find matplotlib's subplot useful.

Part 2 (3 pts)

Implement a function that computes the network below.



The o 's here should simply be linear combinations of the x 's (that is, the activation function in the output layer is the identity). Supecifically, use $o_i = \sum_j w_{ji} x_j + b_i$. Include the listing of your implementation (i.e., the source code for the function; options for how to do that in LaTeX are here) in your report for this Part.

Part 3 (10 pts)

We would like to use the sum of the negative log-probabilities of all the training cases as the cost function.

Part 3(a) (5 pts)

Compute the gradient of the cost function with respect to the weight w_{ij} . Justify every step. You may refer to Slide 7 of the One-Hot Encoding lecture, but note that you need to justify every step there, and that your cost function is the sum over all the training examples.

Part 3(b) (5 pts)

Write vectorized code that computes the gradient with respect to the cost function. Check that the gradient was computed correctly by approximating the gradient at several coordinates using finite differences. Include the code for computing the gradient in vectorized form in your report.

Part 4 (10 pts)

Train the neural network you constructed. Plot the learning curves. Display the weights going into each of the output units. Describe the details of your optimization procedure.

Part 5 (10 pts)

The advantage of using multinomial logistic regression (i.e., the network you constructed here) over using linear regression as in Project 1 is that the cost function isn't overly large when the actual outputs are very far away from the target outputs. That causes the network to not adjust the weights too much because of a single training case that causes the outputs to be very large. You should come up with a dataset where the performance on the test set is better when you use multinomial logistic regression. Do this by generating training and test sets similarly to how we did it in lecture . Show that the performance using multinomial logistic regression (on the test set) is substantially better. Explain what you did and include code to clarify your point. Clearly explain why your experiment illustrates your point.

Part 6 (10 pts)

Backpropagation can be seen as a way to speed up the computation of the gradient. For a network with N layers each of which contains K neurons, determine how much faster is (fully-vectorized) Backpropagation compared to computing the gradient with respect to each weight individually. Assume that all the layers are fully-connected. Show your work. Make any reasonable assumptions (e.g., about how fast matrix multiplication can be peformed), and state what assumptions you are making.

Part 7 (20 pts)

I am providing the TensorFlow code for training a single-hidden-layer fully-connected network here . The training is done using mini-batches. Modify the code to classify faces of the 6 actors in Project 1. Use a fully-connected neural network with a single hidden layer. In your report, include the learning curve for the test, training, and validation sets, and the final performance classification on the test set. Include a text description of your system. In particular, describe how you preprocessed the input and initialized the weights, what activation function you used, and what the exact architecture of the network that you selected was. Experiment with different settings to produce the best performance, and report what you did to obtain the best performance.

Use 30 images per actor in the test set.

Unlike in Project 1, you must remove non-faces from your dataset. Use the SHA-256 hashes to remove bad images. You may additionally hardcode the indices of the images that you'd like to remove.

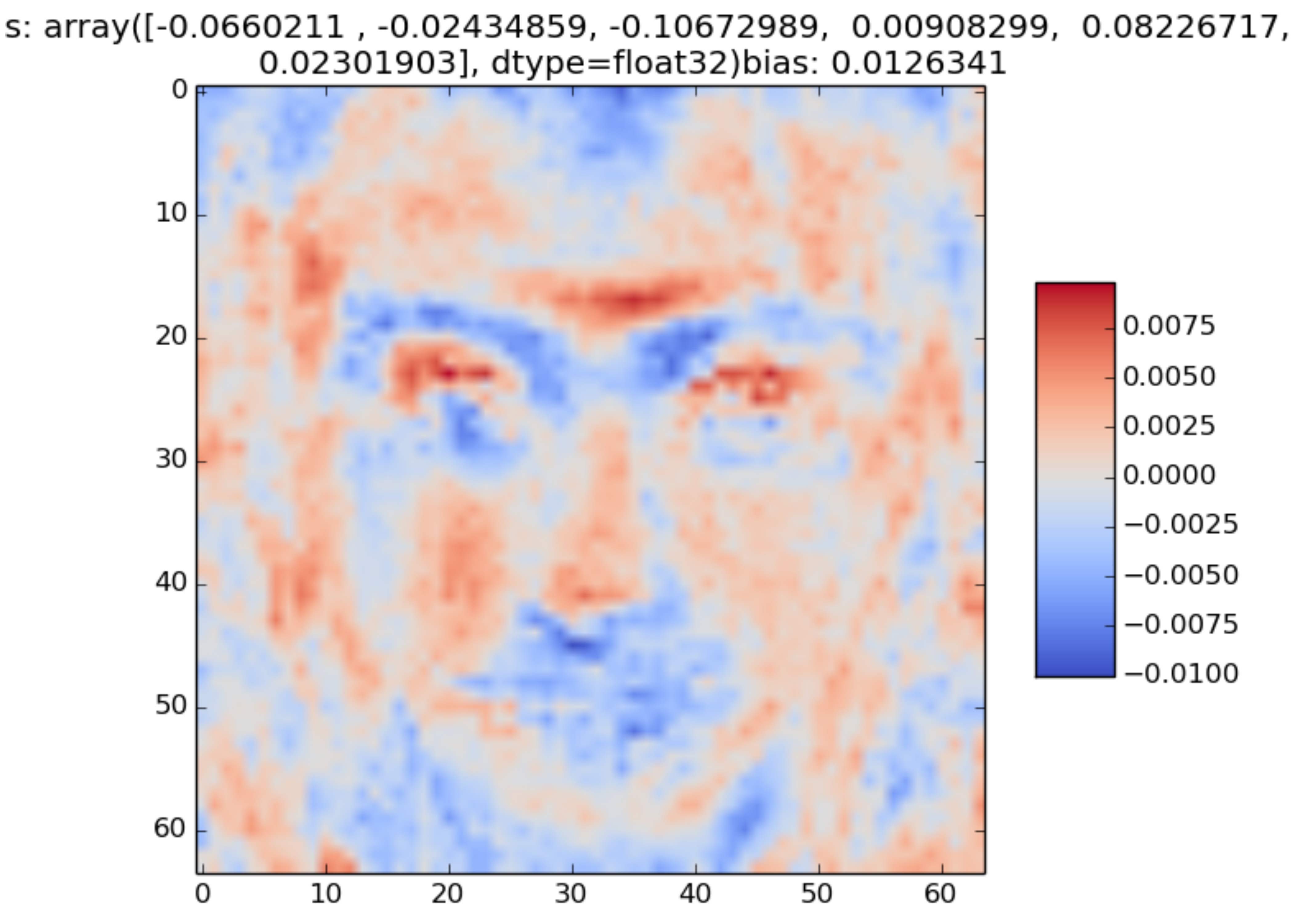
Part 8 (5 pts)

You are not required to use regularization in Part 7 (although that would produce nicer weight visualization). Find a scenario where using regularization is necessary in the context of face classification, and find the best regularization parameter λ .

Part 9 (10 pts)

Select two of the actors, and visualize the weights of the hidden units that are useful for classifying input photos as those particular actors. Explain how you selected the hidden units.

A sample visualization is below.


```
s: array([-0.0660211 , -0.02434859, -0.10672989,  0.00908299,  0.08226717,
          0.02301903], dtype=float32)bias: 0.0126341
```

Part 10 (20 pts)

I am providing the code for AlexNet here .

Extract the values of the activations of AlexNet on the face images. In your report, explain how you did that. Use those as features in order to perform face classification: learn a fully-connected neural network that takes in the activations of the units in the AlexNet layer as inputs, and outputs the name of the person. In your report, include a description of the system you built and its performance, similarly to Part 7. It is possible to improve on the results of Part 7 by reducing the error rate by at least 30%. I recommend starting out with only using the conv4 activations.

Part 11 (BONUS 10 pts)

Produce an interesting visualization of the hidden units that were trained on top of the AlexNet conv4 features.

What to submit

The project should be implemented using Python 2 or 3 and should be runnable on the CS Teaching Labs computers. Your report should be in PDF format. You should use LaTeX to generate the report, and submit the .tex file as well. A sample template is on the course website. You will submit at least the following files: `faces.py` , `digits.py` , `deepfaces,py` , `deepnn.tex` , and `deepnn.pdf` . You may submit more files as well.

Reproducibility counts! We should be able to obtain all the graphs and figures in your report by running your code. The only exception is that you may pre-download the images (what and how you did that, including the code you used to download the images, should be included in your submission.) Submissions that are not reproducible will not receive full marks. If your graphs/reported numbers cannot be reproduced by running the code, you may be docked up to 20%. (Of course, if the code is simply incomplete, you may lose even more.) Suggestion: if you are using randomness anywhere, use `numpy.random.seed()` .

You must use LaTeX to generate the report. LaTeX is the tool used to generate virtually all technical reports and research papers in machine learning, and students report that after they get used to writing reports in LaTeX, they start using LaTeX for all their course reports. In addition, using LaTeX facilitates the production of reproducible results.

Using my code

You are free to use any of the code available from the CSC411 course website.

Readability

Readability counts! If your code isn't readable or your report doesn't make sense, they are not that useful. In addition, the TA can't read them. You will lose marks for those things.

Academic integrity

It is perfectly fine to discuss general ideas with other people, if you acknowledge ideas in your report that are not your own. However, you must not look at other people's code, or show your code to other people, and you must not look at other people's reports and derivations, or show your report and derivations to other people. All of those things are academic offences.