STAT 946 Deep Learning

Project Title: Applying Convolutional Neural Networks to Simultaneous Localization and Mapping

Project Number: 1

Group Members:

Surname, First Name	Student ID	Your Department
Aubin, Sean	saubin	Systems Design Engineering
Komer, Brent	bjkomer	Systems Design Engineering

One of our group members is taking STAT 841 (Classification)?

- × No
- □ Yes

Your project falls into one of the following categories. Check the boxes which describe your project the best.

1. □ **Kaggle project.** Our project is a Kaggle competition.
   - Our rank in the competition is ... .
   - The best Kaggle score in this competition is ... ..., and our score is ....
2. × **New algorithm.** We developed a new algorithm and demonstrated (theoretically and/or empirically) why our technique is better (or worse) than other algorithms. (Note: A negative result does not lose marks, as long as you followed proper theoretical and/or experimental techniques).
3. × **Application.** We applied known algorithm(s) to some domain.
   - □ We applied the algorithm(s) to our own research problem.
     - □ We used an existing implementation of the algorithm(s).
     - □ We implemented the algorithm(s) ourself.
   - × We tried to reproduce results of someone else's paper.
     - □ We used an existing implementation of the algorithm(s).
     - × We implemented the algorithm(s) ourselves.

Our most significant contributions are (List at most three):

(a) Showed that GoogleNet or whatever network currently performs best on ImageNet is ideal for localization applications

(b) Compared two version of filters

(c) Showed improvement in detection by averaging over the results from different network architectures and withing layers

1 Introduction

The objective of Simultaneous Localization and Mapping (SLAM) is to map an unknown environment while simultaneously localizing where an agent resides in that map. An essential part of algorithms that attempt to solve this problem is visual feature extraction. Usually, this is approached by using SIFT or SURF to generate local features to be used to create a bag-of-words representation of the image. However, as demonstrated in great success of applying Convolutional Neural Networks (CNNs) to image classification problems, CNNs present an attractive alternative to typical SLAM feature extraction. In this report, we replicated and expand upon the results of the paper "Convolutional Neural Network-based Place Recognition" [1], which we will refer to from now on as "the reference paper". We re-developed the filtering algorithm and implemented an alternative using Numpy [8]. In terms of CNNs, we replicated the results in the paper using the Overfeat [5] CNN architecture. We also explored alternative architectures using Caffe [2], including GoogleNet [7], a CIFAR10 example included in Caffe, VGG [6] and AlexNet [3]. The code for this report can be found at `https://github.com/Seanny123/DeepSLAM/`.

2 Experimental Setup

2.1 Summary of Reference Paper

The reference paper used a two-step process to test their SLAM approach. First, they create a confusion matrix of the features extracted from the dataset (as in Figure 1), then they filter the confusion matrix to find the most likely location for each frame. A good way to think about the testing method conceptually, in particular the confusion matrix, is to imagine they are testing a robot who has already built a database of video frames by going over a path. This original set of frames forms the x-axis of the confusion matrix. Now, the robot is revisiting segments of that path (either in the same order or reverse) and we want to check that using the video feed alone we can determine where we are in the original path. Thus, every new frame adds a new entry into the y-axis of the confusion matrix.

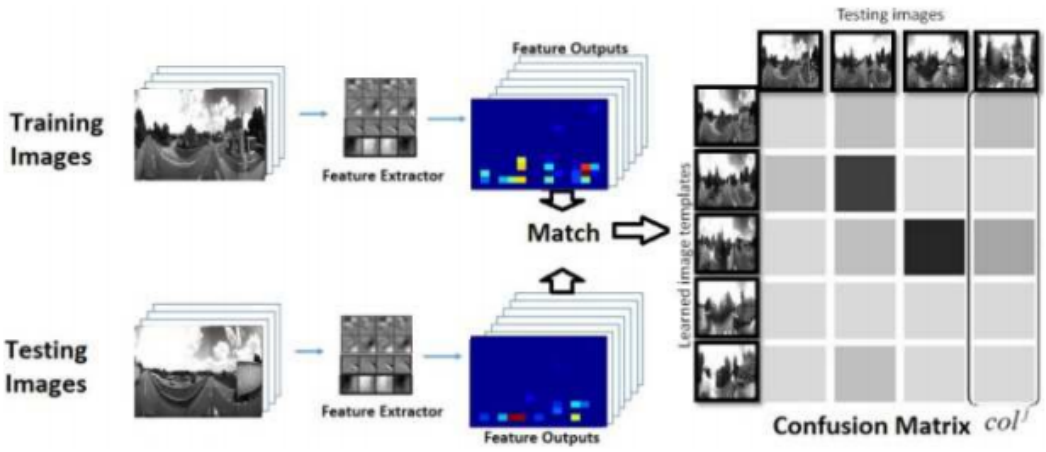


Figure 1: How the confusion matrix is constructed. Courtesy of the reference paper.

Although it is tempting to just select the frame with the highest similarity in that column,

instead a filter is created to take into account the velocity of the robot and the potential for jitter. This helps restrict the results found to be smooth and consistent. For example, if the robot is estimated to be in a particular place in one frame, it does not make sense for the robot to suddenly be in a far away and unconnected location in the next frame.

Jitter was eliminated by eliminating all cells that varied too much from their neighbours.

An estimation of the velocity (how quickly the robot is progressing through the pictures) is done by doing linear fitting of the last few valid cells. Only cells within a tolerance of the slope pertaining to the velocity estimate are selected, as seen in Figure 2.

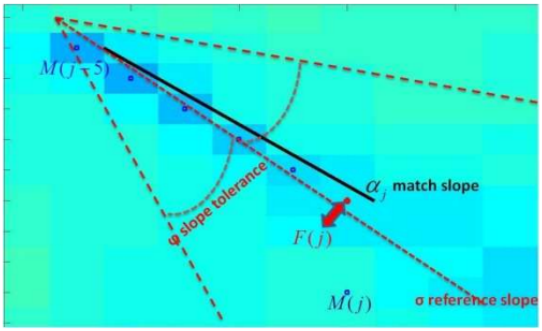


Figure 2: How velocity is used to eliminate outliers. Courtesy of the reference paper.

A more in-depth description of the constants used, as well as mathematical formulations of this filter can be found in the reference paper.

2.2 Filter extension

Although not described in the paper, the Matlab code acquired from the authors used an extension to the filter to achieve their results. This Boosted Filter traded off precision for recall by filling in all previously unfilled confusion matrix cell values for a given frame size. See Figure 3 for a more intuitive explanation. In simple terms, once a valid cell is found according to the simple filter, all cells previously skipped because they did not meet the criteria of the simple filter are then assigned the index of the valid cell. Additionally, all cells within the aforementioned frame detected in the future will be assigned the valid cell value unless the simple filter over-writes it. In other words, if a frame didn't find a match, but within a certain frame size a match is found, it is assumed that the robot was immobile the whole time. It is also assumed the robot is immobile until proven otherwise.

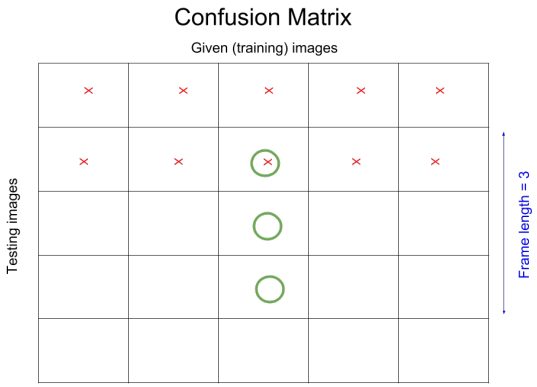


Figure 3: An example of the boosted filter with a frame length of 3, scanning from top to bottom. After a valid cell is found in the third row (marked by a purple circle), the second and fourth row is assumed to have the same index (marked by green circles). This corrects the previously skipped second column (marked by red crosses).

The lack of intuition for this filter is probably why it was not included in the reference paper.

2.3 Comparison of CNN Architectures Used

The CNN architectures are chosen due to their performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [4]. AlexNet won the 2012 challenge for classification, Oxford's VGGNet won the 2014 challenge for localization, GoogleNet won the 2014 challenge for classification and OverFeat won the 2013 challenge for localization.

Due the unconventional architecture of GoogleNet, we did not iterate over all the layers and instead only iterated each "DepthConcat" layer.

Additionally, as a means of connecting with our lab's research and exploring the implications of compressing the resolution of the images, a CIFAR10 architecture was used. This architecture lends itself well to soft-LIF non-linearities (see Figure 4), which allow for implementation of the CNN as a spiking neural network and thus able to run on low-power neuromorphic hardware, as demonstrated by Eric Hunsberger (in press). Although we were able to implement a soft-LIF in Caffe, we were unable to use it for training, thus it's results will not be discussed in this report.

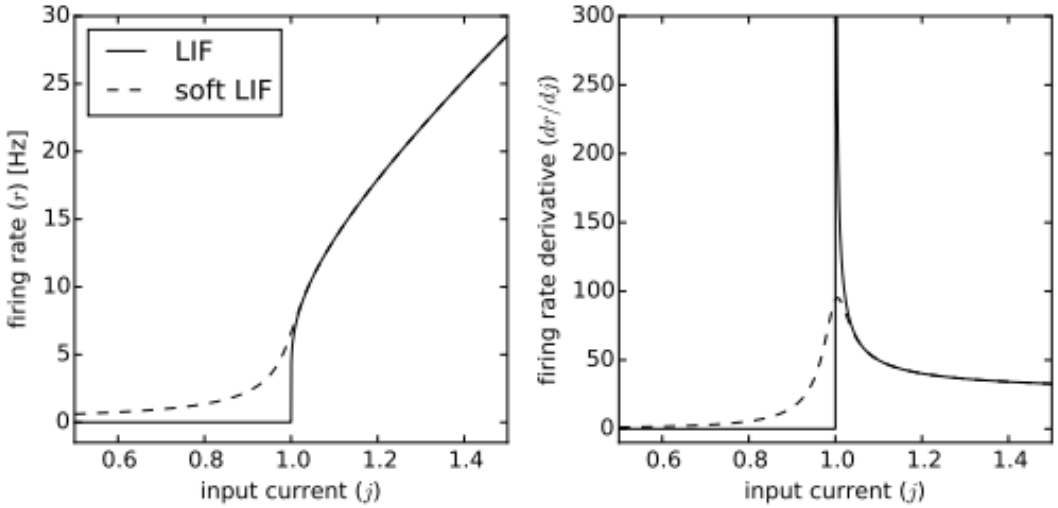
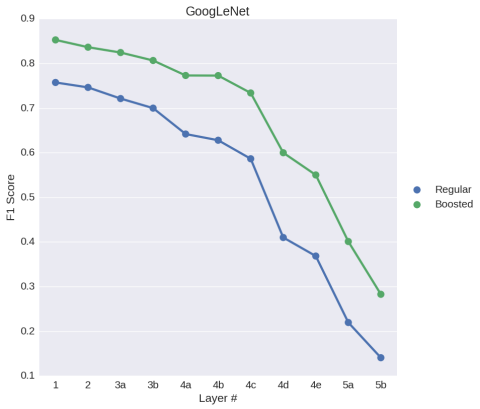


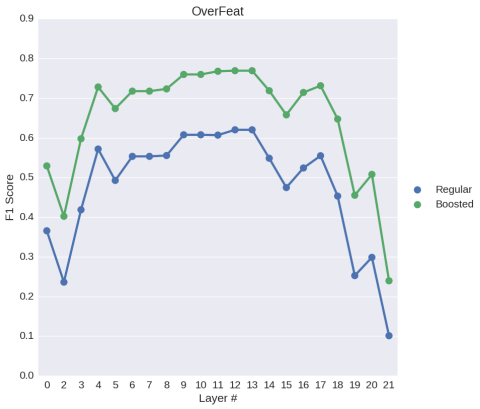
Figure 4: A comparison of the activation function of a LIF-neuron and it's soft-LIF approximation. Note the derivative of the soft-LIF is smooth allowing it to be used for back-propagation. Figure courtesy of Eric Hunsberger.

3 Results

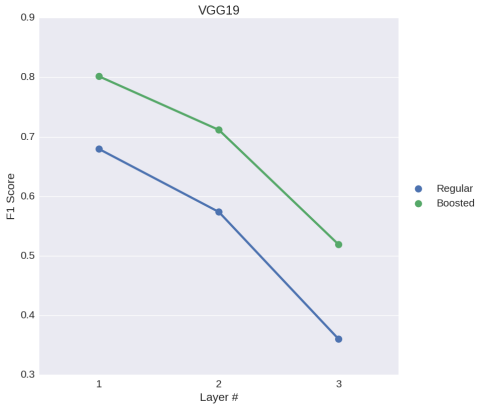
Overall, the networks who performed better on the ImageNet classification challenge performed better in the SLAM test as well, as demonstrated in Figure 5. Obviously, the compression required by the CIFAR10 dataset greatly reduced it's performance. Consequently, if spiking neural networks were to be investigated for this application, they would first have to be applied to a more modern CNN architecture.



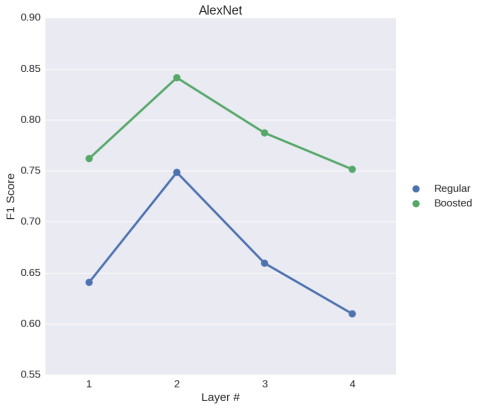
(a) GoogleNet



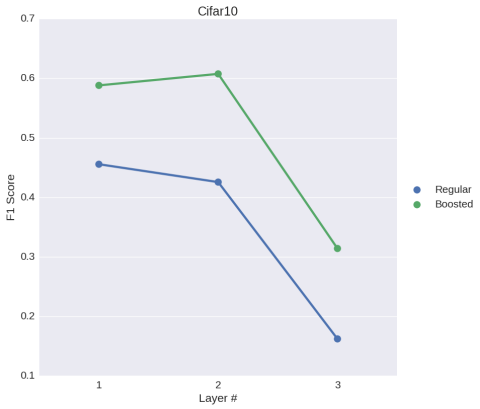
(b) Overfeat



(c) VGG19



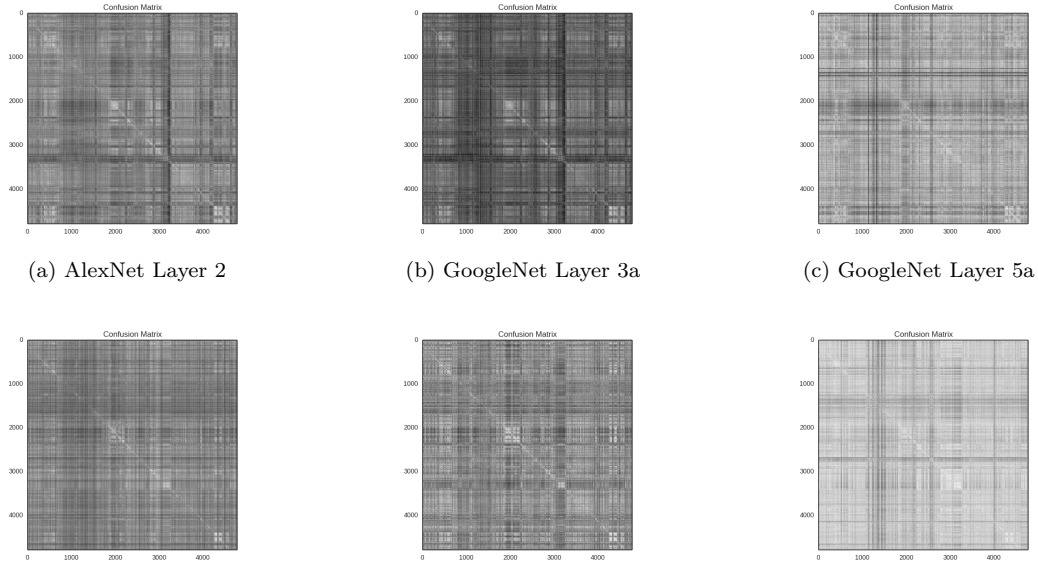
(d) AlexNet



(e) CIFAR10

Figure 5: F1 metrics for all CNN architectures. The performance for all networks decreases with later layers, since they are over-trained on the ImageNet task. Note that few layers were tested for VGG19 (the 19 Layer implementation of Oxford's VGGNet) due to computational constraints in both time and memory.

Given the wide variety of performance and by observing visual differences in how the confusion matrices looked between networks and network layers, we tried to further increase precision and recall by constructing a confusion matrix from the average of the confusion matrices found for each layer of the network. The intuition behind this approach is that when inspecting the confusion matrices visually, a clear line can be seen for where the correct path should be, but there is a lot of noise in the image. Some example confusion matrices from different layers and networks can be seen in Figure 6. The noise seems to be somewhat structured, with the structure differing between different layers of the same network, and differing more dramatically across layers of different networks. If the noise is not correlated across layers, averaging the outputs of the layers together will effectively 'blur' much of this noise away and allow the effects from the correct path to be seen more clearly.

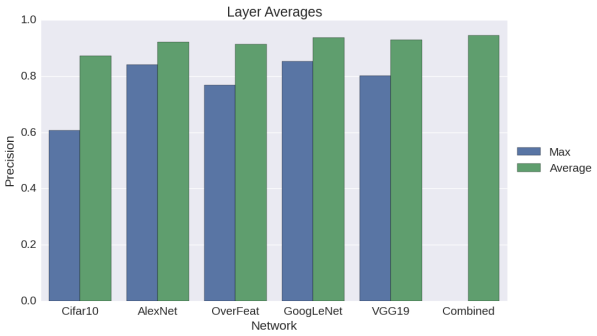


(a) AlexNet Layer 2 (b) GoogleNet Layer 3a (c) GoogleNet Layer 5a

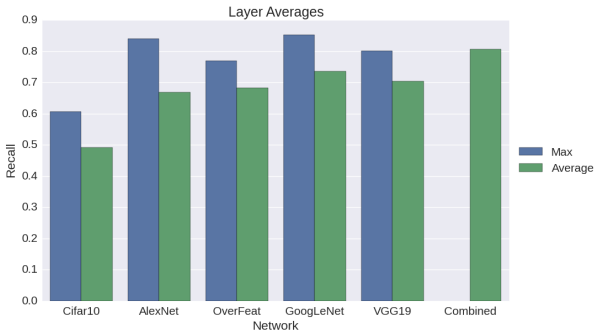
(d) OverFeat Layer 5 (e) OverFeat Layer 11 (f) OverFeat Layer 17

Figure 6: Visualization of Confusion Matrices

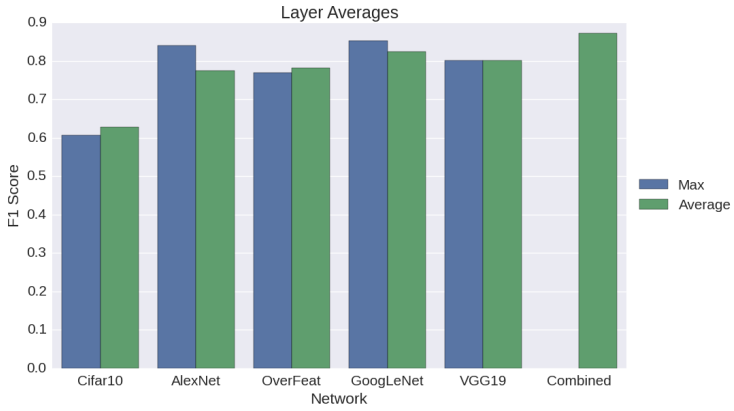
Another way to think of this is that each layer is better at representing different types of features that are useful to the task, while simultaneously having some features that are not helpful. Averaging multiple layers allows the less useful features to have a smaller effect, while the good features have a correlated output and as a result still have a strong effect. This is analogous to ensemble methods. The results of this new technique are shown in Figure 7. These three results were chosen to be combined because they are the best results out of those particular networks and each network has a very different architecture. Adding in Cifar10 (the problem of low resolution) or VGG19 (too similar to GoogleNet) reduced the performance.



(a) Precision



(b) Recall



(c) F1

Figure 7: Metrics of the various network architectures, both maximum performance and average performance of all the layers.

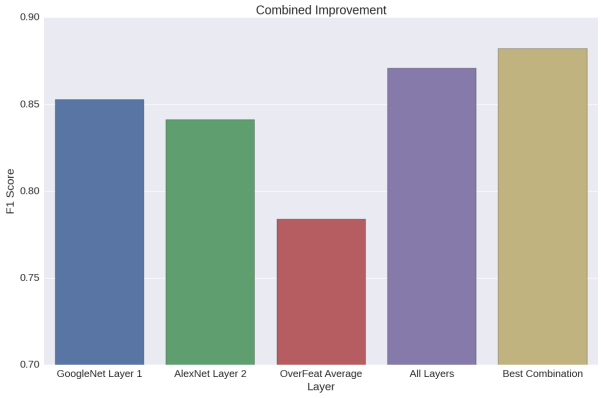


Figure 8: Further improvement found by combining particular layers with good results. 'All Layers' refers to averaging the matrices from every layer except those in the Cifar10 network. 'Best Combination' is the average of the first layer of GoogleNet, the second layer of AlexNet, and the average result of all layers of OverFeat.

All of the Caffe-based CNN architectures were able to run fast enough on a GPU (all-layers) to enable real-time operation (processing more than one image a second). Overfeat (due to the fact that the code we acquired could not be run on the GPU) was slower than realtime, however in the reference paper it was already verified that a Caffe implementation (which we could not locate) could run in real time. The code for the OverFeat implementation that we ran can be found here: `https://github.com/sermanet/OverFeat`. Further computational savings could be made by only running the relevant layers.

4 Discussion

As seen in the previous section, the newest classification based network, GoogleNet outperformed all the other networks. Additionally, when augmented with the confusion from other networks, it performed even better. Consequently, if one were to continue to pursue this task, we would recommend to use whatever network has won the most recent ImageNet competition in the domain of classification. However, it should be noted that this suggestion comes with one caveat, given the bizarrely stellar performance of layer 2 of the AlexNet. In this report, we've assume this to be an outlier, however further investigation of a wider variety of networks and datasets could prove otherwise. As a final recommendation, combinations of unrelated nets should be pursued further for better results, assuming a platform where computation power and power supply are available in ample supply.

References

[1] Zetao Chen, Obadiah Lam, Adam Jacobson, and Michael Milford. Convolutional neural network-based place recognition. *arXiv preprint arXiv:1411.1509*, 2014.

[2] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[5] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[8] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.