Locality through Modular Network Topology

Oliver Lipton and Harry Huchra

December 18, 2017

Abstract

Our experiment concerns designing a robot to accomplish the task of locality grounding. Through a dataset built by its own sensors, the robot would attempt to classify a sensor reading from somewhere in its environment and identify where in the environment that sensor reading was taken from. If this task could be done both accurately and precisely, the robot could effectively develop a grid system to define its coordinates in the environment. This has many applications: a robot able to determine its position after a period of exploration could be further trained to navigate anywhere in its environment. We found that, while the robot's accuracy scaled poorly to the number of possible locations, it was generally off by at most a single location, guessing a sector adjacent to the correct one. Furthermore, we experiment with different ways that sensor readings could be combined to best succeed at this task. We determine that attempting to train a network that takes both sensor and camera inputs from scratch was cumbersome, slow, and difficult. We had better results training networks to use visual and sonar data individually, and then combining their outputs as inputs to a third network. This modular approach caused a notable increase over either of the individual networks, and trained faster and more accurately than the combined, branched network.

1 Introduction

Classification is a quintessential task in machine learning. After the explosion of popularity of Convolutional Neural Networks, classification of images has become a popular domain for experimentation. However, most of these experiments have dealt with large datasets of images taken from outside sources. This works for machine learning as a whole, but in the domain of robotics, we were interested in seeing an agent that does classification over a dataset accumulated from its own sensors. We came up with a simple task: a robot would explore a small environment, and with a dataset built from its explorations, learn to classify its location. In theory, this would let the robot to ground the world into a map representation that would allow it to plot efficient routes via higher-level thinking.

1.1 Related Work

Most of the work done in deriving locality from sensor data has been in the field of large scale mobile robotics. Accordingly, research has mostly been conducted in real world environments with a much larger scope in mind. Having a robot locate its position within a floor map of an office building, or even within a city, for example. Past work in this area has been centered around probabilistic Monte Carlo analysis, and often avoids using visual data as input due to its noisiness and inherent unpredictability [5]. Work done on learning through vision usually focused instead on a Visual Bag of Words approach. In this approach, images are divided up into distinct visual features -"words"- and locality is predicted based on the number of relevant words in the image that correspond to a particular location. For instance, one experiment [1] used this approach to test whether a robot could localize itself within the streets of cities in India based on images taken from the roof of a moving car. As others have demonstrated, features used in Visual Bag of Words must be consistent across images taken from multiple angles and distances, therefore scale invariant features are a necessity [4].

This approach naturally lends itself to neural networks, since networks are able to identify the necessary visual words as features and classify them accordingly. While this approach to locality has been implemented with neural networks before on real world data [2], we wanted to experiment

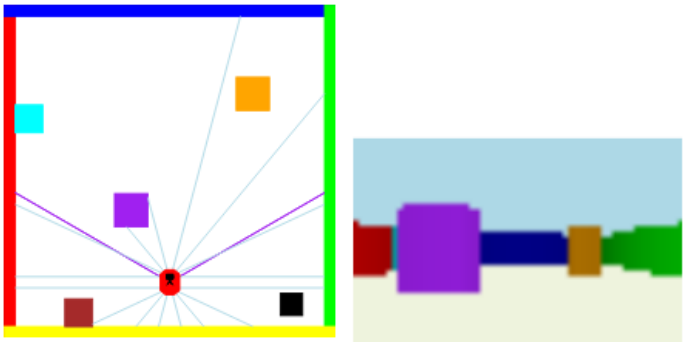

Figure 1: Our simulated environment for the robot and a sample of its camera feed.

with it in a less chaotic environment. Rather than use statistical sensor fusion [3] to blend our robot's sensors, we opted to create a neural network capable of choosing weights for the sensors independantly.

2 Experiments

2.1 Virtual Environment

We used the Python Jyro simulator to model a virtual Pioneer robot with a camera and sonar sensors. We created a virtual environment for the robot that was visually distinct. The robot was equipped with a camera and 16 sonar sensors. We designed the environment to make the classification task as easy as possible. Since the robot would be using a camera, we color coded the environment to make visual input less ambiguous. Since the robot would be using sonars, we placed the objects in the environment asymmetrically, so as to reduce the ambiguity of sensor readings.
The environment is divided into a 6x6 grid. The robot's goal will be to take input from its sensors at some place in the environment, and from that input, indicate which sector it's in that space on that grid.

2.2 Datasets

We created large and small versions of two separate datasets for comparison. The first, which we nicknamed the "Hand of God" dataset, was generated by repeatedly moving the robot to a random point and random heading. Then, if the robot was not overlapping with a wall, taking a sensor reading, and matching it to the robot's sector. The small version of this dataset was made with 5000 random placements and contained approximately 4,300 valid entries, while the larger version was made with 10,000 random placements and contained approximately 8400 valid entries.

The second was called the "Wandering" dataset. This was generated by allowing the robot to motor-babble for a while, taking sensor readings and filling the dataset as it did. The motor-babbling made sure not to crash into walls, but had no incentive to explore or curate a dataset in any way. Large and small versions of this dataset were also made using 5000 and 10000 robot positions respectively.

There are many pros and cons to both datasets. The Hand of God dataset is, arguably, cheating: by moving the robot artificially between each capture, we are violating the capabilities of a theoretical automatous robot, and getting results that could not be replicated in a physical environment. This partially violates our intention to make the robot perform this task through its own means. However, this method was able to create a balanced dataset, with each sector represented a reasonable number of times.

On the other hand, the Wandering dataset did not involve as much human intervention, and was in a way more legitimate. However, due to the nature of the obstacles, motor-babbling did not

result in the robot exploring the entire environment. This resulted in an irreparably unbalanced dataset, as large regions of the map were not represented. It is arguable if this is a problem or not: on a theoretical physical robot, we would not expect it to understand areas it had not explored, so building a dataset through exploration should be sufficient.

2.3 Neural Network topology

The area we experimented the most was in the topology of the neural network we used for classification. Originally, we had three basic neural networks we experimented with:

The first focused entirely on the robot's camera. We used a convolutional and fully connected neural network constructor that took the camera feed as input, and outputted a onehot encoding of the sector it predicted the robot was in, resulting in 36 output nodes.The network contained two convolutional layers for feature extraction and used 2x2 max pooling layers after each layer to reduce computation time. The output of the second convolution/pooling layers was passed to two fully connected hidden layers before the output layer.
The second focused entirely on the sonar sensors. It took 16 inputs, one for each normalized sonar input. Like the vision network's hidden nodes it used a fully connected neural network architecture and outputted a onehot encoding of the sector it predicted. After some preliminary trials we found that using four fully connected hidden layers for this network gave us good results without causing over-fitting. Both this network and the former -naive networks without sensor blending- were refined over multiple trials using the smaller datasets.

The third network was a combination of the first two: it took camera input on one branch and sonar input on the other. Each branch had all the same layers as the first two networks minus their outputs. The branches then combine into a single network, which outputs the onehot encoded prediction. We hypothesized that this network would score better than either of the previous networks thanks to learning correlations between the vision and sonar data. We also hypothesized that these correlations would be best learned by a network which trained on both sets of data at once rather than learned from them separately.

To test this, we created a fourth network built on top of the vision and sonar networks. Once these first two networks were trained, they were run on every point in the dataset, and their outputs became the inputs for a new dataset. This smaller network was first trained to predict the robot's location first on those outputs alone, and then later based on their trained hidden layers. We expected this fourth network to make somewhat less accurate predictions than the blended network. We theorized that since back propagation would occur independently for each set of sensor data, the network wouldn't be able to learn correlations between sensors in the same way.

Our original network topologies are shown in Figure 2, and the layout of the fourth network is diagrammed in Figure 3. Each network/dataset pair was trained until the network's validation accuracy plateaued sufficiently that it neither rose nor fell by no more than five percent over fifty successive epochs and one percent over the course of twenty successive epochs. Once each network was trained, we examined its predictions by manually giving it input locations (particularly locations that we considered difficult to classify) that were not included in its dataset and observing its predictions. For each network, we also determined the average error in the euclidean distance between its prediction and the corresponding grid square of the virtual environment. This average error was computed for each prediction in the training and test sets.

3 Results

3.1 Variance of Results by Dataset

Training the networks on the smaller versions of each dataset did not show many meaningful differences compared with training them on the larger datasets. For example, trained on the smaller Hand of God dataset, the networks achieved 65 percent validation accuracy for vision, and 27 percent for sonar. Training again on the larger dataset achieved 71 percent and 40 percent respectively. This demonstrates that making correct predictions based on sonar readings was helped

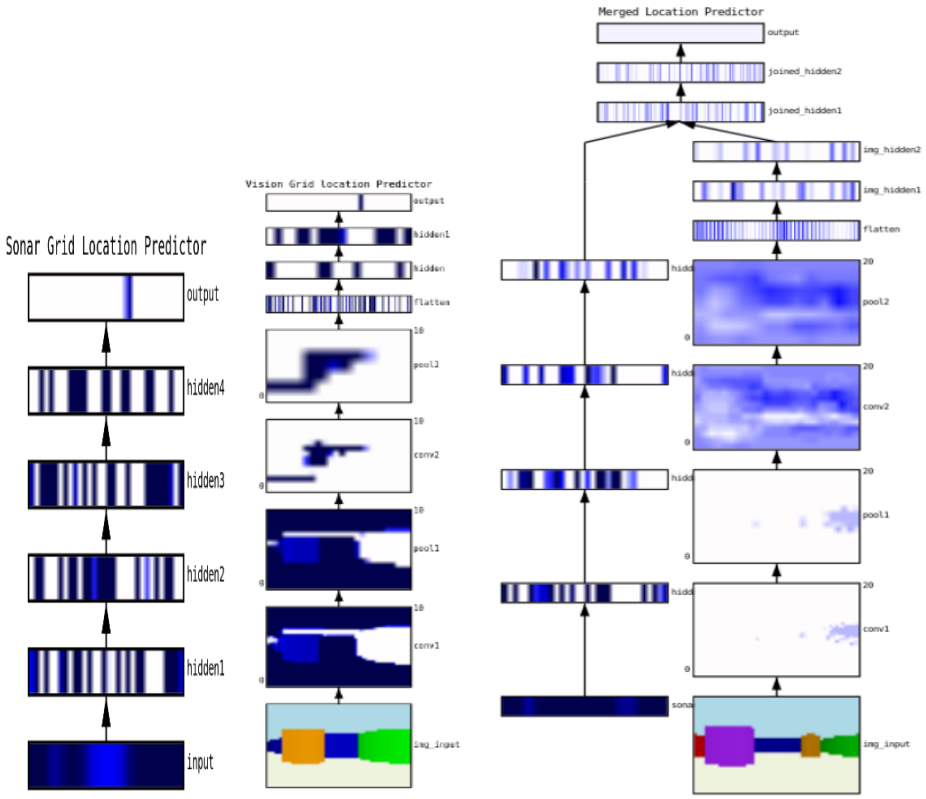

Figure 2: The topology of our original networks. From left to right: The sonar network (32 nodes per hidden layer), the vision network (40 nodes per hidden layer), and the blended network (140 nodes per blended hidden layer).

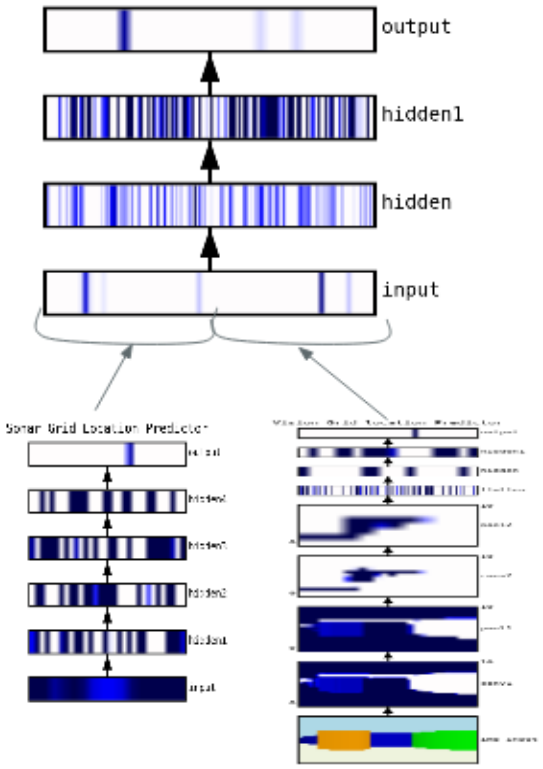

Figure 3: The smaller prediction network used to test the output of the pre-trained vision and sonar networks. Each of its input nodes is an output node or hidden node in the previous networks, and like the blended network its hidden layers each have 140 hidden nodes.

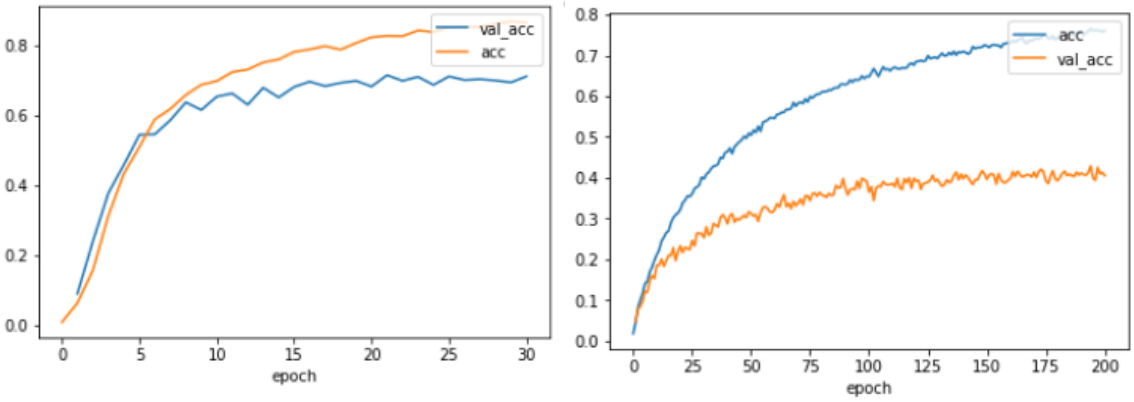

Figure 4: Vision and Sonar Network Results on the "Hand of God" dataset.

more by extra data than making correct predictions based on vision. However, this is somewhat expected given the fact that the robot's camera images already contain far more information than its sonar readings do. Ultimately, the large datasets are more useful for judging the best possible performance of each network, even if the smaller datasets are more applicable to a real robot learning to navigate a real environment.

Something of greater note is that the accuracy of the networks trained on the Hand of God data set is noticeably worse than the 83 percent/45 percent accuracy achieved by the networks trained on the larger "Wandering" dataset. Since the robot has been programmed with a tendency to avoid walls while wandering, it actually travels to fewer locations. It therefore makes the locality task easier for itself by cutting down on the number of output locations it needs to predict from its testing data. In this sense its much more applicable to a real robot than the first dataset is.

However, the accuracy achieved when training with the former is retained when performing locality predictions while testing the network with locations from a robot given a different basic wandering pattern. The networks trained by the Wandering dataset are not quite so robust as they first appear, as they achieve a much lower success rate for locations that they never got close to during training.

3.2 Results of the Individual Sonar and Vision Networks

As expected, our naive sonar and vision networks both performed much better than chance on all of our datasets.

While the vision network performed significantly better than the sonar net, vision alone was unable to find a completely accurate representation of the environment even when trained on the larger datasets. Yet although its predictions are seemingly not accurate 100 percent of the time, the vision network trained on the larger datasets learns an extremely accurate *general* representation. When examined more closely, the network's predictions are accurate to within two grid squares 100 percent of the time, and within 1 grid square roughly 90 percent of the time. This means the network is in one sense completely accurate, as it leans to differentiate general regions of the world.

3.3 Results of the Combined Networks

Like the naive networks, both the blended network and the smaller combined network also learn representations that can make locality predictions which are usually only off by a few squares. However, their average error is substantially lower. As hypothesized, training a network from a blending of both sonar and vision data achieves a better accuracy on all four datasets than either of the naive networks alone. As seen in Figures 5 and 6 the combined network is able to achieve a validation accuracy of 72 percent/87 percent on the Hand of God and Wandering datasets respectively.

However, this network is actually noticeably less accurate than the smaller network trained on the outputs and hidden layers of the naive networks. These are able to reach -at best- 77 percent and 90 percent validation accuracy for the respective datasets. The blended network comes pretty

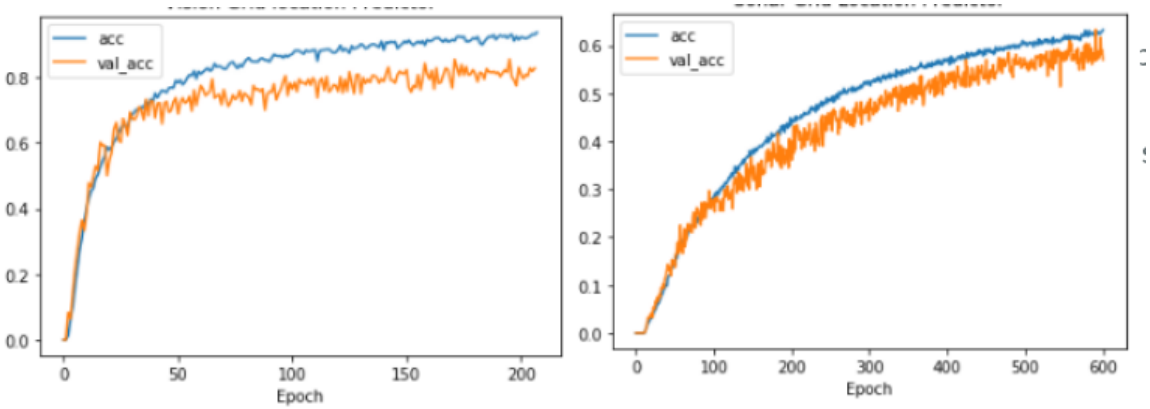

Figure 5: Vision and Sonar Network Results on the "Wandering" dataset.

close in the case of the latter, since training the smaller network on only the outputs of the naive networks (without the hidden layers) only achieves 88 percent accuracy. However the blended network's output is still much noisier and still takes much longer to converge than the simple network does.

3.4 Euclidean Evaluation

One flaw with our results so far is that they treat the task as a straight-forward classification problem: if the network does not output the correct prediction for its location, it is marked incorrect, regardless of how close or far its prediction is. Since the boundaries of the grid are arbitrary, it is reasonable that the robot would struggle on said boundaries, and so being off by one sector is much less problematic than being off by several. With this in mind, we created a Euclidian Evaluation function, that grades the network based on an average of how close its guess is to the correct sector at each point in the dataset. A correct prediction would be scored as 100%, a prediction that was at the polar opposite side of the area would be scored at 0%, and anything in between scaled appropriately. The results were very optimistic: every network, including the sonar-only network, scored above 95% on this evaluator. This shows that, indeed, while some of the networks are often wrong, they are generally off by no more than three sectors. This means that the network is better equipped to the task than the results above suggest.

4 Discussion

In one sense our hypothesis was partially correct. Using both visual data and sonar data to perform localization is more accurate than using either one alone. However, training a network to learn from both at the same time was not only an extremely time-consuming process, but arguably a less effective one as well. The smaller network trained on the outputs and hidden layers of the first two networks is a better classifier in every way.

Training on the smaller datasets, the smaller network learned representation of the world in about 20 epochs. The blended network takes more than ten times as long to learn a comparable representation, and its predictions are still less accurate than those of the smaller net. The predictions of the two networks are much closer together when they're trained on the larger datasets, which suggests that the blended network *can* eventually learn a good representation given enough data and enough time.

Its theoretically possible that if we'd created even larger datasets, the blended network might have been able to surpass the smaller network when trained on those. But its worth asking the question whether collecting a dataset *even larger* than 8400 entries is really worthwhile. A robot navigating a small space such as the one we've designed ideally shouldn't need that many data-points to begin localizing itself. Indeed, other localization techniques which don't use neural networks are able to converge to an answer relatively quickly compared to our approach. For this reason it would make much more sense to use our original networks and small combination network

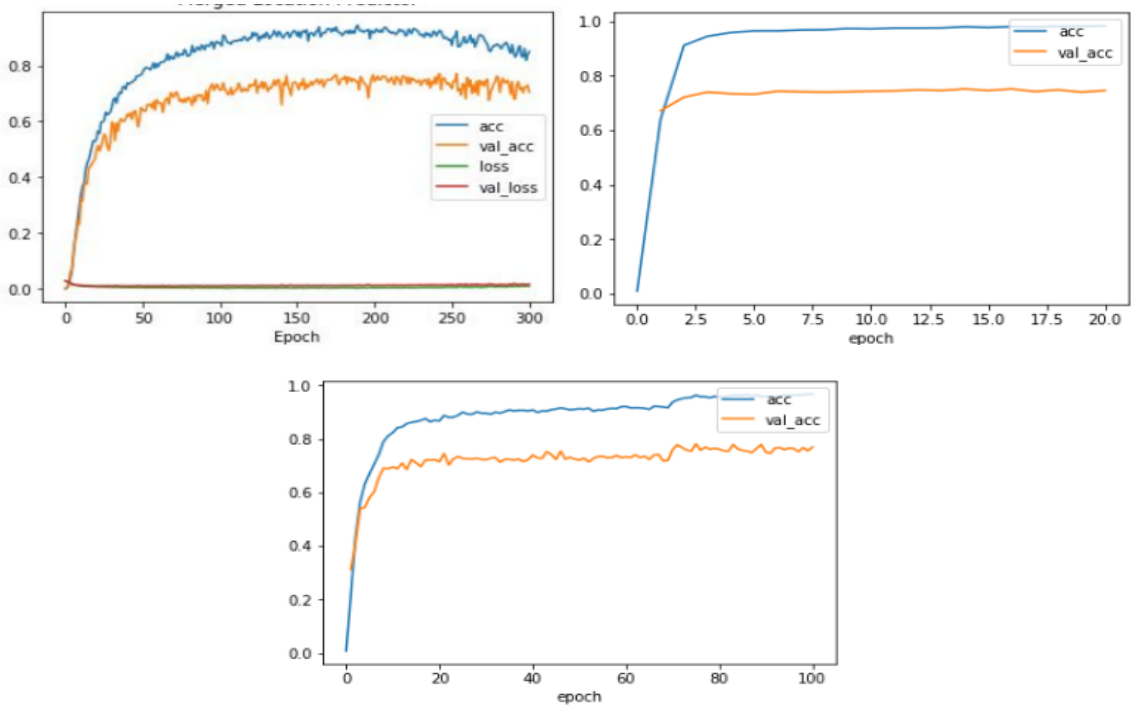

Figure 6: Blended Network and Smaller Combined Network Results on the "Hand of God" dataset. Top left: Training accuracy (blue) and validation accuracy(orange) of the blended network. Top right: Training and validation accuracy of the modular combined network trained on the outputs of the naive networks. Bottom: Accuracies of the modular network when trained on the last hidden layers of the naive networks.

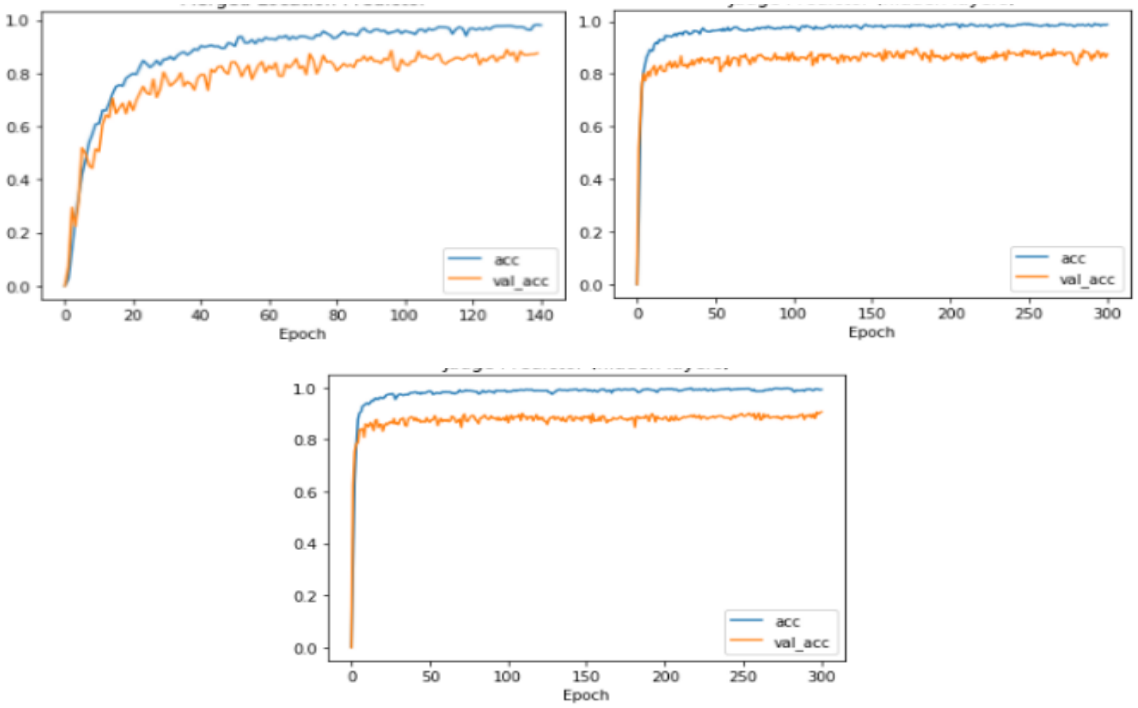

Figure 7: Blended Network and Smaller Combined Network Results on the "Wandering" dataset. Top left: The training accuracy (blue) and validation accuracy(orange) of the blended network. Top right: The training and validation accuracy of the modular combined network trained on the outputs of the naive networks. Bottom: The training accuracies of the modular combined network trained on the last hidden layers of the vision and sonar only networks.

as part of a practical navigation system.

The difference in performance between the smaller network and the blended network is probably due to the unfortunate reality that processing visual data is very different from processing sonar data. As the success of the smaller combination network proves, there are strong correlations between the two correlations between the two. But these are correlations related only to the locality of the robot and not to any other related features that the blended network searches through. For example, a direct relationship between how blue a wall is and what sonar readings it produces may exist in our simulated environment, but knowing this is not actually that useful in the locality task. A much more effective metric to learn would be something like the aspect ratio of objects in the camera feed vs. the sonar readings, but the blended network won't find this right away. It has a much larger search space to go through, whereas the smaller network is limited to a smaller search space which happens to include the most efficient answers.

5 Conclusions and Future Work

From our experiments, we are pleased with our robot's ability to accomplish the task. Unfortunately, due to the poor scaling of the neural network using onehot encoded outputs, it does not scale well to higher precision or larger environments. We ran a few experiments on a larger grid (24 x 24) and the results were terrible. However, when using a 6x6 grid, the robot not only correctly identifies its position a majority of the time (except for the sonar-only network), but when it is off, it is almost always off by only one or two sectors. We are equally pleased with the results we achieved through the modular network architecture: because we were able to achieve greater results by combining two less effective networks, there is a good chance that this same strategy could scale to other problems. There are many possible directions for future work. We would like to experiment with increased use of modularity: instead of just separating out networks for vision and sonar, we could separate out networks that are trained to deal with particular smaller areas of the grid, or networks trained to identify other features of the environment. Additionally, the robot could use the relatively high accuracy of its locality grounding to learn to navigate the environment. There is still work to do in order to increase precision without losing accuracy: the rate at which the network grows to accommodate a larger number of possible sectors is not sustainable, so another strategy is called for. Our training method and network topologies could be further improved by replacing the onehot encoded outputs with continuous x and y coordinate outputs, and using a Manhattan Distance loss function. If this works, the network could scale much better and incorporate the gradual distance loss function into training, which could help with more complex environments. However, it would also need to overcome the difficulties of continuous outputs. Our experiments have demonstrated, however, that with relatively little exploration beforehand, robots can quite easily learn to identify their location in their environment.

6 References

References

[1] S. Achar, C. V. Jawahar, and K. Madhava Krishna. Large scale visual localization in urban environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5642–5648, May 2011.

[2] J. A. Janet, M. W. White, T. A. Chase, R. C. Luo, and J. C. Sutton. Pattern analysis for autonomous vehicles with the region- and feature-based neural network: global self-localization and traffic sign recognition. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3598–3604 vol.4, Apr 1996.

[3] M. Kam, Xiaoxun Zhu, and P. Kalata. Sensor fusion for mobile robot navigation. *Proceedings of the IEEE*, 85(1):108–119, Jan 1997.

[4] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 2051–2058 vol.2, 2001.

[5] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.