Intermediate Supervisors for Confidence Evaluation and Error-Guided Learning

Matthew Kowal
Ryerson University
350 Victoria St, Toronto, ON M5B 2K3
matthew.kowal@ryerson.ca

Abstract

Predicting the errors that Deep Neural Networks will make and understanding the confidence of their predictions are of critical importance when implementing them into real world systems. This project shows that a separate network (supervisor) can take the intermediate activations of a deep network (student) and predict the pixel-wise errors that the deep network will make for the application of semantic segmentation with an accuracy of 39.2% mean-IOU. These intermediate supervisors can also be used to represent the confidence in a student's prediction and are shown to be marginally correlated with the accuracy of the student's output. One attempt to use the intermediate supervisors to increase the baseline accuracy of the student network was made but was unsuccessful, though alternative strategies are discussed. The code can be found at: www.github.com/MKowal2/supervisor-error-guide

1. Introduction

Deep Neural Networks (DNNs) and in particular Convolutional Neural Networks (CNNs) have in recent years proven to be extremely effective for solving problems in many areas of computer vision [3]. A wide selection of challenges ranging in difficulty have been approached using CNNs, including object detection, object tracking, semantic segmentation, depth estimation and 3D scene recreation from a single image [1][5, 7][11, 13]. As the uses for these algorithms multiply, their real-world implementation in many areas, such as autonomous vehicles, security systems, and healthcare, mean that society is increasingly dependent on their proper functioning. While dependancy on neural networks is beneficial for the most part, there are situations in which the decisions made by DNNs can lead to tragic consequences. For instance, in May 2016, the world experienced the first death due to a mistake in an assisted driving system [15]. It is clear that these networks will be more heavily relied upon in the near future and need to be not only more accurate, but more confident in their predictions.

Many different techniques have been shown to successfully represent the uncertainty of a network's prediction. Most existing approaches use a built in regressor which can show some measure of confidence in the output of the network. I tackle a similar problem but with a separate network altogether whose sole purpose is to predict the error of another network. I show in this project that it is possible to train this 'supervisor' network on the intermediate activations from a 'student' network trained on semantic segmentation in order to predict the errors the student will make in its final prediction. I also argue that introducing an error-prediction term in the loss function can guide a network to learn new features, creating a new loss function that combines the original loss as well as a uncertainty term determined by the supervisor network. This both encompasses uncertainty in the output as well as leveraging the features learned in the mid-layers of the network in order to improve its accuracy.

2. Methods

This section first reviews related work involving uncertainty, confidence, and intermediate supervision. I then define semantic segmentation which is the task used in this project. I formalize the types of error used as well as the objective of the intermediate supervisors. I review the loss functions implemented to achieve the desired learning and then finally define the type of confidence metric used in this project.

2.1. Related Work

The subject of confidence is a wide area of research where the objective is to associate each output with a pixel-wise measure of the probability of that pixel being correct [6]. In [10], they put a Gaussian distribution prior over the weights and output of a model to represent uncertainty associated with each model output. It is shown that including this uncertainty in the loss function correctly can improve baseline methods in semantic segmentation by 1-3%. In [12], they present a supervised approach for confidence pre-

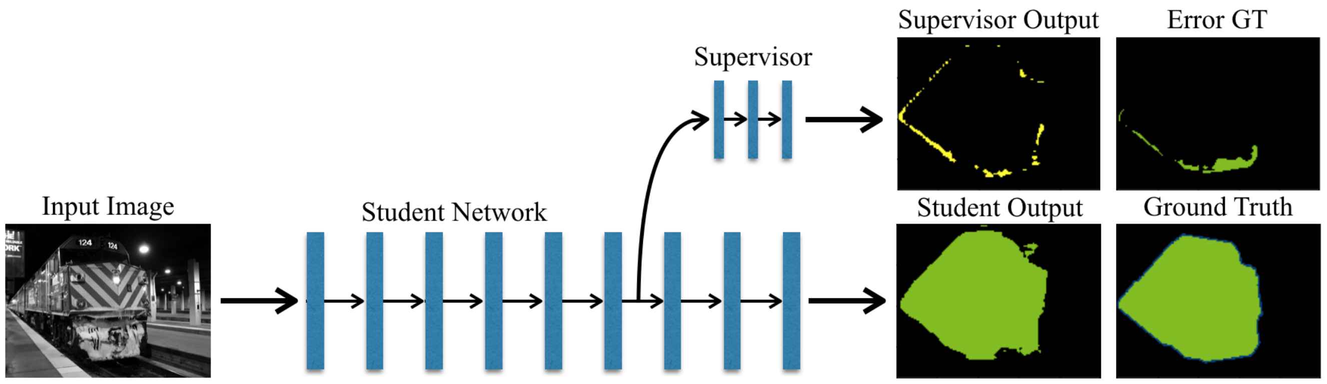

Figure 1. The architecture for a single supervisor network predicting the error of a student network. The supervisor takes the intermediate activations of a particular layer of the student network as input and outputs the error mask for the same example.

diction for optical flow, by estimating if a particular pixel is likely to be far from the true End Point Error (EPE). The most closely related work to my project that I know of is in [8], where, for the task of stereo matching, they train a random-forest classifier to predict whether or not a particular pixel's disparity prediction is likely to be correct.

Other strategies frequently used to increase the accuracy of DNNs include auxiliary classifiers [14], holistically-nested outputs [16], or other types of intermediate losses [9]. The motivation behind these intermediate stage tools involves issues with backpropagation, in which earlier layers have difficulty learning due to the signal from the loss function saturating as it travels through the network from the output. One can also see, particularly in [16], that the network becomes much better at predicting the ground truth in the later layers.

In a sense, the proposed project combines the two ideas above in order to increase the accuracy of a semantic segmentation network. This is done in two steps: (1) I show that information can be learned by using a separate neural network trained to predict confidence in the form of pixel-wise error; and (2) this information can be combined in the form of a loss function which can guide the segmentation network to learn a new set of features that is more robust to pixels that are normally hard to classify.

2.2. Semantic Segmentation

The goal of semantic segmentation is to outline and label all relevant objects in an image. Formally we can define this: given the kth image x_k from a dataset $X = \{x_k | \, k = 1, ..., N\}$, label each pixel $x_k^{(i,j)}$ (i being the pixel rows and j being the pixel columns) as the correct categorical object (person, road, plane, car, etc). One can represent a neural network as a function f which takes as input an example from a dataset and outputs this segmentation as:

$$f(x_k) = \hat{y}_k \tag{1}$$

where $\hat{y}_k$ is the prediction of the network compared against the ground truth for that example $y_k \epsilon Y = \{y_k | \, k = 1, ..., N\}$. The comparison is formulated as a loss function between the prediction and ground truth. It is minimized with respect to θ, the parameters of f:

$$\min_{\theta} L_{NN}(\hat{y}_k, y_k) \tag{2}$$

using the gradient of the loss function as a measure of how much to update each parameter through a process called backpropagation. This is repeated iteratively until L_{NN} is stable at a minimum which achieves good results for the task at hand.

2.3. Error Masks as Ground Truth

Consider a prediction $\hat{y}_k$ for a semantic segmentation problem. The ideal solution is for our network to learn a mapping that labels every pixel value correctly, $\hat{y}_k^{(i,j)} = y_k^{(i,j)}$ for all (i, j). A pixel-wise error mask has multiple formalisms and is a method of capturing the errors in a single example. For the kth example image, we define the class-dependent binary error mask to be:

$$E_{k-binary}^{(i,j)} = \begin{cases} 0, & \text{if } \hat{y}_k^{(i,j)} = y_k^{(i,j)} \\ 1, & \text{otherwise} \end{cases} \tag{3}$$

Similarly, we define the class-independent binary error mask to be:

$$E_{k-indep}^{(i,j)} = \begin{cases} 0, & \text{if } \hat{y}_k^{(i,j)} \text{and } y_k^{(i,j)} > 0 \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

Finally, we define the multi-class error mask to be:

$$E_{k-multi}^{(i,j)} = \begin{cases} y_k^{(i,j)}, & \text{if } y_k^{(i,j)} \neq 0 \text{ and } \hat{y}_k^{(i,j)} \neq y_k^{(i,j)} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Equation 3 and 4 are unique in the way they handle classes. In equation 3, any pixel that is not the same class is treated as an error while in equation 4, pixels that are labeled as a different class but still as an object (not background) are labeled as correct. This binary error serves as a useful intermediate step in order to better understand the features when training a network to predict the error mask as the output.

Finally, equation 6 is a multi-class error mask. In the error mask, every pixel that is incorrectly labeled will take on the value of the ground truth pixel's value. For example, if an image of a horse had one pixel on the horse labeled as a dog in the output, this pixel would take on the class value of 'horse' in the corresponding ground truth mask.

2.4. Error Mask Prediction Using Intermediate Activations

For a semantic segmentation network, such as the Deeplab V2 ResNet-101 [2], the activations of a layer are the values outputted from the non-linearity portion of a specific layer (ResNet-101 has 101 non-linear layers). For the mth non-linear layer and the kth example from the dataset, we denote these activations as A_k^m. The first objective of the project is to design and train a neural network, f_{side}^m that can accurately predict the error mask using the intermediate activations from layer m as the input. After dropping the example reference subscript for readability, this can be written as:

$$f_{side}^m(A^m) = \hat{E} \tag{6}$$

where $\hat{E}$ is the prediction of the error mask for one example. We then want to minimize some loss $L_{side}^m(\hat{E}, E)$ to find the optimal parameters of our network f_{side}^m.

2.5. Loss Function

The loss function used in the DeepLab network is the pixel-wise cross entropy loss:

$$L_{CE} = -\sum_{c=1}^{M}\sum_{i,j} y_{0,1} logP(y^{i,j} = c) \tag{7}$$

where c is the class, $y_{0,1}$ is the indicator function which is 1 when the pixel is classified correctly and $P(y^{i,j} = c)$ is the probability of $y^{i,j}$ being in class c. Initially, the supervisor network signal was added in as a loss term using the binary cross-entropy loss:

$$L_{sup} = \sum_{i,j} logP(y^{i,j} = 1) - \sum_{i,j} logP(y^{i,j} = 0) \tag{8}$$

This term penalizes the student when the supervisor is incorrect in its error prediction. The issue with this loss, I discovered, is that it does not necessarily guide the student to be more accurate in its segmentation. It merely updates the weights in such a way that the *supervisor* is more accurate at predicting error regardless of how much error there actually is. I found that in some examples, the student would sacrifice accuracy in order for the supervisor to be more accurate.

To enforce that the networks learns weights that motivate the supervisor to make *less* predictions overall (meaning less error), a supervisor regularization term λ_{sup} is added:

$$\lambda_{sup} = \frac{number\, of\, errors\, predicted}{number\, of\, pixels} \tag{9}$$

Finally, a second regularization term, called starting point regularization L_{SP} is also added so that the fine-tuned network does not stray far from the original weights, which are quite good at semantic segmentation already:

$$L_{SP} = ||w - w^0||_2^2 \tag{10}$$

where w^0 are the original weights of the DeepLab network trained on PascalVOC and w are the new weights that are updated through backpropagation through each iteration. Without this term, it is possible for the network to find a solution that does not take into consideration the previous training as it has no motivation to keep the weights from changing too much. This term was crucial for removing various irregularities during the error-guided training process. The final loss function for the error-guided fine tuning process is:

$$L = L_{CE} + \gamma_1 L_{sup} + \gamma_2 L_{SP} + \gamma_3 \lambda_{sup} \tag{11}$$

where γ_1, γ_2 and γ_3 is the weighting parameter for the supervisor loss, starting point regularization, and supervisor regularization term respectively.

2.6. Confidence

The outputs of the supervisors are discrete values. For the binary cases, each pixel is labeled as a 0 (background) or a 1 (error). In the multi-class case, each pixel is valued between 0 and 21, where each value represents an error for that specific object. The difficulty in measuring confidence is how to formulate the discrete output as a continuous one that can represent confidence.

The measure of confidence I used was to take the overall ratio of non-error pixels to the total number of pixels for each image:

$$C = \frac{n_{cor}}{n_{tot}} \tag{12}$$

where n_{cor} is the number of predicted correct pixels and n_{tot} is the total number of pixels. This essentially means that the more pixels that are predicted to be errors, the lower the confidence, and vice-versa.

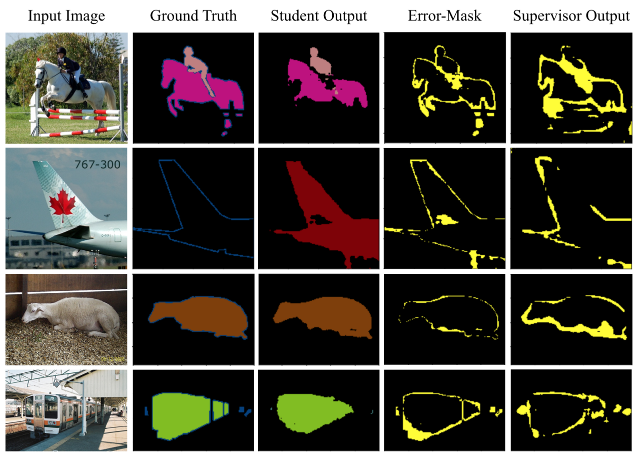

Figure 2. Four example images taken from the validation set with the supervisor trained on the 100th activations. The supervisor output should match the error-mask.

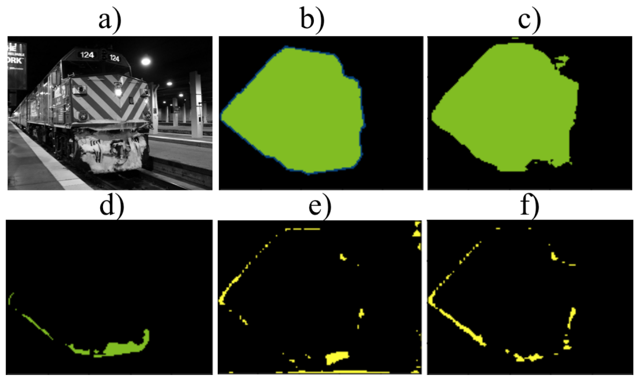

Figure 3. Example showing the effect of padding. a) Input, b) ground truth, c) the output of DeepLab's ResNet-101, d) the error mask (binary + class dependant), e) my networks output with 0-padding (notice the border at the bottom and top right), f) my networks output with reflective padding. Error predictions use 100th layer activations.

2.7. Motivation

The main intuition behind this project is that the set of features that can be learned to predict pixel-wise error in the output may not be the same set of features that are good for semantic segmentation. There is no doubt some overlap between these two sets, but the often made assumption that they are the same has not been verified. If these sets of features are not the same, than it is certainly possible (and even probable) that the accuracy of the student can be improved by leveraging information about features that will contribute to error. This is done by adding an additional term to the loss function which guides the network to: 1) learn features that are 'good' for semantic segmentation, and 2) *not* learn features that can be used to accurately predict errors in its own output. Essentially what this is doing is transforming the loss space so that the learned parameters of the network lie at a better minimum than before.

3. Implementation

The network used is a ResNet-101 network adapted to semantic segmentation and can be found online at https://github.com/speedinghzl/Pytorch-Deeplab [2].

3.1. Class-Balancing

Due to the nature of the dataset, there are vastly more non-error pixels than error pixels for the intermediate network's ground truth, E^m. Without balancing the classes, the network quickly converges to predicting no errors in the entire image. Since over 80% of the pixels are the background class, the trivial solution is to only minimize the parameters with respect to the background class as it provides 80% of the signal through the network. Since the loss used is the cross-entropy loss, each pixel was weighted according to its class in order to penalize these types of solutions.

$$L_{CE}(x) = -\beta \sum_{i,j} logP(y^{i,j} = 1) - (1-\beta) \sum_{i,j} logP(y^{i,j} = 0) \quad (13)$$

where β and $1 - \beta$ is the class weighting for edges and background respectively. I found through experiment that a class weighting of $\beta = 0.15$ works best.

3.2. Padding

As a convolution moves over an image, it is preferable to go beyond the borders of the image, so that the kernel can activate from objects that might be only partially in the image. It is then necessary to provide the kernel with some values for the pixels that are beyond the border. This is referred to as padding.

The code from [2] was initially implemented with zero-padding, where the pixels outside the image borders are black (zero intensity). This was an issue for the network because the contrast between the padding and the border was treated as a segmentable object in some cases. There were many examples with the majority of the border pixels predicted as errors by the intermediate network like in Figure 3. This was fixed by converting the padding from zero-padding to reflective padding, which has little to no contrast relative to zero-padded images.

In order to keep the effects on performance consistent, I compared the overall accuracy of the DeepLab model with mirrored padding after error-guided supervision instead of the original zero-padding. This dropped the performance of

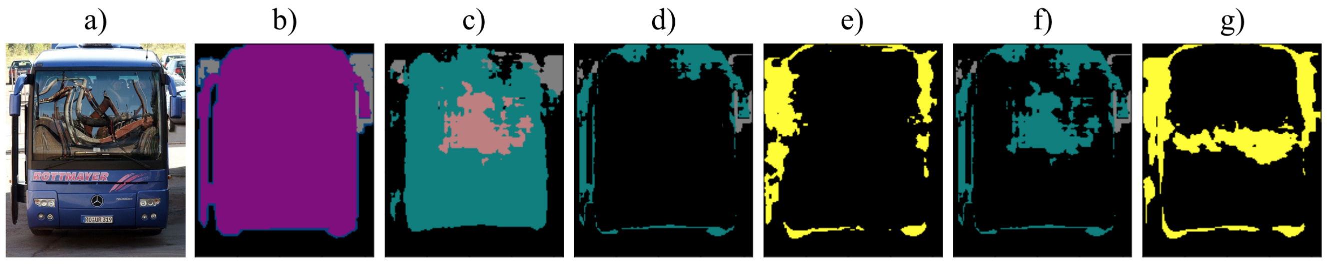

Figure 4. An example showing the difference of being trained on class-dependant vs class-independent error masks. a) input image, b) ground truth, c) output of DeepLab's ResNet-101, d) class-independent error (see equation 4), e) my networks output for class independent error, f) class-dependent error (see equation 3), g) my networks output for class dependent error.

the network from 73.6% to 72.7% mean intersection-over-union (meanIOU).

3.3. Dataset - PascalVOC

The PascalVOC-2012 dataset [4] is a dataset consisting of over 10,000 images for training and validation. It consists of 21 object classes (including the background class) and multiple objects can appear in a single image. Each object is segmented by pixel values of 255 at the edge, and then the class number on the inside of the boundary. The standard metric used for semantic segmentation challenges is meanIOU which is calculated by dividing the intersection of correct pixels predicted by the total number of pixels labeled (ground truth and prediction), for each class.

3.4. Architecture

The binary supervisor was a two layer convolutional network consisting of a 3x3 convolutional layer, a batchnorm layer, a Rectified Linear Unit (ReLU), and finally a 1x1 convolutional output layer. The multi-class supervisor contained three layers. It consists of a 5x5 convolutional layer, batchnorm, ReLU, followed by the same architecture as the binary supervisor for the last two layers. The networks were trained with a learning rate of $2.5e^{-5}$, a batch size of 1, momentum 0.9, and weight-decay of 0.005.

4. Results

The results section is split into four sections. The first part covers the results from the binary cases, both class dependent and class independent. The second section reviews the multi-class error case. I then show the results of the confidence metric as well as how it can be visualized and its accuracy measured quantitatively. Finally, the last part shows the results from the error-guided learning phase where I attempted to improve the baseline accuracy of the DeepLab network with the supervisor applying a feedback signal to the network.

4.1. Binary Error Prediction

Figure 2 shows some examples from the binary case. Recall, we have two instances of the binary error-mask case: class-independent error and class-dependent, corresponding to equations 4 and 3 respectively. The error masks for these two methods only differ when there are overlapping objects of different classes. For the 100th layer, the independent-error case and dependent-error case had a meanIOU accuracy of 37.6% and 39.2% respectively on the PascalVOC validation set.

One issue regarding the accuracy of the binary cases is that the features learned can be activated by any object in the image. This means that regardless of the class, if an object looks difficult to segment, according to the network, then it will predict errors in those areas even though the pixels are the background class. This is very apparent in figure 5 where a slide is predicted by the supervisor as an error inducing object. This decreases the accuracy (in terms of meanIOU) and is not very useful for guiding the original network, but it can tell us information about errors that could be made if that object *was* a class in the dataset.

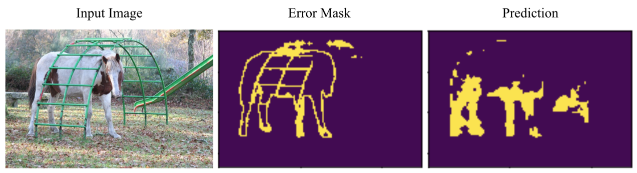

Figure 5. The slide is predicted as a difficult object to segment even though it is not one of the 21 classes.

4.2. Multi-Class Error Prediction

A set of the supervisor predictions for the multi-class case can be seen in figure 6. It is apparent that the network is learning some high-level information about where errors will arise, but it does not perform nearly as well as the binary case in terms of meanIOU or low level detail. A supervisor trained on the 100th layer could only manage 7.1%

Error Type	*Layers*			
	30	50	90	100
Independant	31.2	32.1	32.7	37.6
Dependant	31.6	29.9	32.9	**39.2**
Multi-Class	-	-	-	7.1

Table 1. % meanIOU results for all error mask cases.

accuracy (meanIOU) on the test set, even though some of the outputs are relatively accurate.

The main problem with the class dependent output is the drastic increase in complexity of the task. The supervisor is now tasked with learning class specific features that create error in the output. The issue is that some classes may share features that can create error, such as the limbs of horses and humans being difficult to segment because they are thin. A deeper network should be able to better capture the multi-class nature of the problem, as a three layer network is rarely used for such a large number of classes in a segmentation problem.

In figure 7, the expected pattern resides, where supervisors trained on earlier layers generally have a more difficult time predicting the error. In all three error-mask scenarios the most accurate supervisor is layer 100. The multi-class accuracy from lower layers was negligible and left out of this diagram for readability.

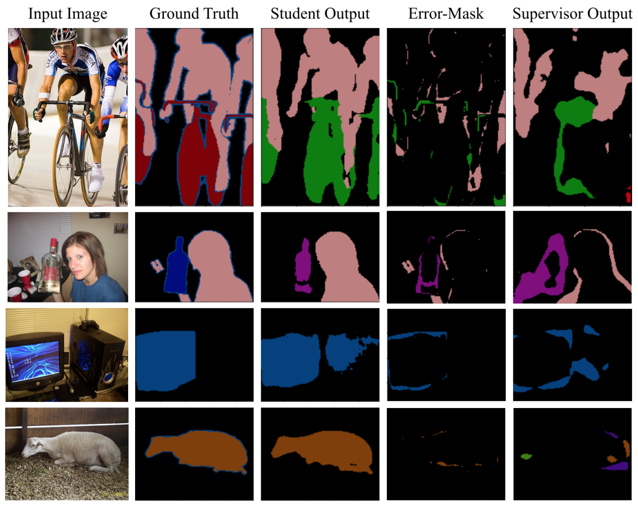

Figure 6. Four example images taken from the validation set with the supervisor trained on the 100th activations. The supervisor output should match the error-mask.

4.3. Confidence Measure

Recall the confidence measure from equation 12. What is desired is a high confidence when the accuracy of the segmentation network is high. However, this will not be the case for every example, as the network can get lucky with

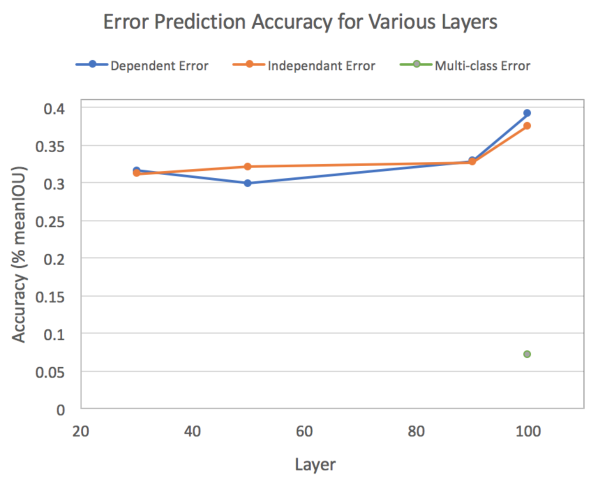

Figure 7. % meanIOU for all three types of error at various layers.

predictions so sometimes the confidence will be low but the accuracy will be high. Regardless, there should be a correlation between the confidence value and the meanIOU of the segmentation network. To quantitatively measure this, I modelled the correlation using linear regression.

It is difficult to compare this to other methods because all other methods that I am aware of view the confidence as a per-pixel regression problem. My solution is per-image which, although has less information overall, can still provide insight into the network's uncertainty in unique ways. Figure 8 shows the correlation between the confidence of supervisor networks against the IOU for each image. Not surprisingly, there is a positive correlation which implies that the more confident the supervisor, the more accurate the student will be.

Table 2 shows the r^2 results for a linear model fit to the confidence vs IOU plots, for multiple layers and error masks. Although the fit is not exceptional, there are interesting trends that show valuable information. The first thing to point out is that the class independent error supervisors have much more correlated confidences than the class dependent error. There are a few possible reasons for this. One is that the network does not care about mislabeled objects and can use more of its parameters to learn class independent features (e.g. edges or occlusions) that contribute more heavily to the errors made by the student. This may suggest that the student's issues revolve more around these types of errors than cross-class errors (e.g. mislabeled objects).

The second thing to note is the 90th layer supervisors perform the best in terms of confidence, followed closely by the 30th layer supervisors. This could be because different layers learn different level features. For instance, the 100th layer might have very high level features learned, while the 90th layer has some more low level information. The 90th

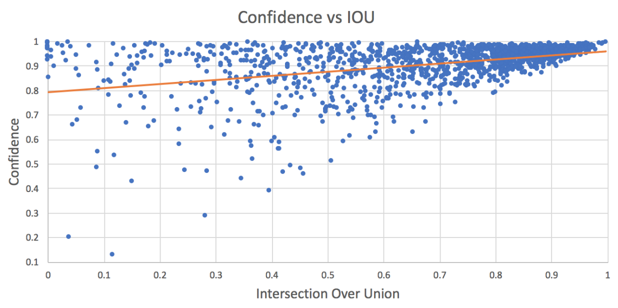

Figure 8. A class independent, 90th layer supervisor's confidence for each image plotted against the students IOU. This contains all examples for the whole validation set.

Layers	30	50	90	100
Independant	0.175	0.143	**0.175**	0.121
Dependant	0.082	0.061	0.084	0.054

Table 2. r^2 results for the confidence vs error plots for both error types. Multi-class was excluded as the confidence was negligible due to the supervisors low accuracy.

layer may predict accurately the *number* of pixels for certain errors (e.g. edges) even if they are mislabelled. This would allow for the 90th layer's confidence value to be a better estimate even though the 100th layer's meanIOU is higher.

4.4. Error-Guided Learning

Unfortunately, the error-guided learning did not perform as intended. I show the fine tuning results in table 3 and then discuss why the process failed as well as what can be done to overcome these hurdles. The learning rate was set to $2.5e^{-7}$ with a batch size of 1 for 20,000 iterations on the PascalVOC training data.

Weighting			meanIOU (%)
$\gamma_1 = 1$	$\gamma_2 = 0$	$\gamma_3 = 0$	55.24
$\gamma_1 = 1$	$\gamma_2 = 1$	$\gamma_3 = 0$	54.95
$\gamma_1 = 1$	$\gamma_2 = 0$	$\gamma_3 = 1$	65.40
$\gamma_1 = 0.3$	$\gamma_2 = 1$	$\gamma_3 = 0$	65.14
$\gamma_1 = 0.1$	$\gamma_2 = 1$	$\gamma_3 = 0$	62.16

Table 3. % meanIOU results for various weightings of the loss function for the error guided learning process. Baseline accuracy to improve is 72.7% meanIOU.

The fine tuning process lowers the meanIOU in all cases, however we can still extract useful information from the results. In general, adding the starting point loss helped and only lowered the accuracy by 10% or less. The best weighting for the supervisor loss appears to sit around $\gamma_1 = 0.3$. More work will be done to figure out the exact reason for failure, however hyperparameter tuning appears to play a large role.

5. Conclusion

Despite the error-guided portion not achieving the desired result, a new method for predicting error for semantic segmentation networks has been developed. This method is not limited to DeepLab's segmentation network but can theoretically be applied to any task in which an error mask can be calculated.

The segmentation supervisors can be trained on three types of error: binary class dependent, binary class independent, or multi-class. The multi-class case is a much more difficult problem but still shows promising qualitative results even though the meanIOU is quite low, at just over 7%. The error predicting supervisors achieve an accuracy of 39.2% meanIOU for the class dependant case.

The error prediction can also be used as a confidence indicator. The intuition is that the more errors that are predicted in an image, the less confident the student network is in its prediction. I showed that the best supervisor has a confidence prediction which correlates with the meanIOU with an r^2 value of 0.175.

6. Future Work

The supervisors performed well at predicting both cases of binary error, however higher accuracy is still possible. Growing the network size could potentially increase the accuracy however one must be careful of using a network that is too big which can overfit and predict errors where there are none. There were a few cases like this, where the network would predict errors on regions of the image without class objects (see figure 5), which is due to the class invariant nature of supervisor.

For the multi-class error prediction, it could certainly still be possible to obtain more accurate results using a supervisor network. The network I used was very small, only three layers, for a fairly complicated task. I will grow the size of the network and hopefully the ability to predict error will improve as the numbers of parameters increase. This is especially the case when using the supervisor on the activations of earlier layers which, as discussed, could not be done with the three layer network.

The confidence metric shows promising results even though the correlation is fairly weak. I expect that as the accuracy of the supervisor networks increase, this correlation will increase as well which will allow the confidence to be a far more accurate tool to determine the uncertainty of the student network.

Due to the timing of the project, and the required time to set aside for report writing, I was not able to run as many experiments as necessary to get the results desired for the error-guided learning process. It is possible, however, that this method won't result in a higher baseline accuracy. Regardless, I plan to run many ablation studies in order to figure out the optimal hyperparameter settings, as well as search for different regularization and loss terms that could improve this process.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.

[3] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks—a review. *Pattern recognition*, 35(10):2279–2301, 2002.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5431–5440, 2016.

[6] Y. Gal. Uncertainty in deep learning. *University of Cambridge*, 2016.

[7] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[8] R. Gouveia, A. Spyropoulos, and P. Mordohai. Confidence estimation for superpixel-based stereo matching. In *3D Vision (3DV), 2015 International Conference on*, pages 180–188. IEEE, 2015.

[9] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5300–5309. IEEE, 2017.

[10] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow. Learning a confidence measure for optical flow. *IEEE transactions on pattern analysis and machine intelligence*, pages 1–1, 2012.

[13] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[15] B. Vlasic and N. E. Boudette. 'self-driving tesla was involved in fatal crash,'us says. *New York Times,¡ day*, 30, 2016.

[16] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

7. Differences Between Courses

The difference is basically the computer vision project assumes that we have the binary error supervisor and then extends its use to confidence, multi-class scenario and error-guided learning. The neural information processing is the part one to this project. It explains in more detail how the binary error prediction works and the theory behind the results.

7.1. Deep Learning in Computer Vision

Questions answered in this project: - How can this error prediction be used to measure confidence? - How do we know it is a good metric? - How does the loss function effect the performance of the network? - Detailed loss functions including class balancing. - 0-padding and how it effects the error prediction/ confidence evaluation. - Binary AND Multi-class cases and how it effects supervisor guidance.

7.2. Neural Information Processing

Questions answered in Neural Information Processing: - How do we learn error? - What features give rise to error? - How do different networks compare for different layers and different error masks? - What is the underlying theory that creates differences in the learning hierarchy? - Why do certain errors arise binary cases? Why are the cases different?