Stacked generative adversarial networks for learning additional features of image segmentation maps

by

Matthew Burke

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Ali Jannesari, Major Professor
Zhu Zhang
Guang Song

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

This thesis culminates my academic experience at Iowa State, and I would like to thank the many people who helped me during my time here. First, I'd like to thank my parents and brothers for their continued support and confidence in me as I took a leap to switch into computer science from a different background. I'd also like to thank the many professors and advisers who helped me be successful during graduate school as a student, researcher, and teaching assistant. My committee members were all very helpful with exploring and refining a variety of research proposals. My weekly paper discussions with Dr. Zhu Zhang and his group of students was extremely beneficial for engaging in the most current topics and ideas in the field. I'm grateful for the guidance and refinement of my research in discussions with Dr. Ali Jannesari that enabled me to complete this work, and enabled me to collaborate with several other researchers. Finally, I'd like to thank my fellow students. Many late hours were spent in the library with Jansel, Modeste, Waqwoya, and Marios, and their dedication and perseverance has been inspiring. A special thanks to Jansel for his help on this thesis.

ABSTRACT

It has been shown that image segmentation models can be improved with an adversarial loss. Additionally, previous analysis of adversarial examples in image classification has shown that image datasets contain features that are not easily recognized by humans. This work investigates the effect of using a second adversarial loss to further improve image segmentation. The proposed model uses two generative adversarial networks stacked together, where the first generator takes an image as input and generates a segmentation map. The second generator then takes this predicted segmentation map as input and predicts the errors relative to the ground truth segmentation map. If these errors contained additional features that are not easily recognized by humans, they could possibly be learned by a discriminator. The proposed model did not consistently show significant improvement over a single generative adversarial model, casting doubt about the existence of such features.

CHAPTER 1. OVERVIEW

Neural networks used for computer vision tasks such as image classification, image segmentation, and image generation have improved dramatically over the past half decade. Classification accuracy on ImageNet, a major benchmark for computer vision capabilities, has improved from 75% in 2015 to over 88% in 2020. Most of these networks can be broadly classified as convolutional neural networks, which learn patterns in images by adjusting weights of convolutional filters. Although a lot of research has been conducted to study what patterns convolutional neural networks learn, there is still no unified consensus about what exactly these features are [Cammarata et al. (2020)]. Recently, researchers at OpenAI have begun an initiative to “zoom in” on neural networks to discover exactly what each neuron and sub-graph of neurons in some of the most well-known neural networks learn [Cammarata et al. (2020)]. If we can better understand what exactly convolutional neural nets are learning on the neuronal level, perhaps we can better design neural networks to learn those same features in more efficient and effective ways. Perhaps we can also design networks to learn additional types of features.

1.1 Learning More Features

In addition to investigating what convolutional neural networks *do* learn, researchers are also pushing the boundaries of what they *can* learn by continually improving scores on computer vision benchmark tests, and demonstrating entirely new capabilities, such as generating photorealistic images from noise [Karras et al. (2019)]. One approach to improving neural networks is to learn more features. Low-level features in the early layers of convolutional neural networks combine to form higher-level features in later layers, so increasing the number of layers in a network has often improved performance [He et al. (2016)].

1.2 Learning New Features

Computer vision may also be improved by investigating if new types of features exist, and if they can be captured and utilized to improve performance of computer vision networks. The goal of this work is to investigate if a new type of feature can be learned to improve a computer vision task. Specifically, I hypothesize that the errors of image segmentation model predictions have features that can be learned, but have not previously been utilized by any computer vision networks. Improving the performance of an image segmentation network via incorporating these features into the network would provide evidence that these features do in fact exist. I propose a novel method for learning these features, and I evaluate its effectiveness at improving image segmentation accuracy.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Generative Adversarial Networks

The method I propose for learning new features utilizes a generative adversarial network (GAN). Generative adversarial networks were first invented by Ian Goodfellow in 2014 [Goodfellow et al. (2014)]. A generative adversarial network consists of two neural networks, a generator and a discriminator, that are trained in an alternating fashion in order for the networks to improve each other.

The generator learns to model the distribution of a dataset. For example, if the dataset being modeled is a set of images, the generator might take a vector of randomly sampled pixel values as input, and try to produce an image that could reasonably be a member of the dataset. It generates plausible data. The discriminator is a classification network that learns to distinguish between "*fake*" images produced by the generator, and "*real*" images from the dataset of real images.

The generator is trained to maximize the likelihood that the discriminator will predict that its output is real. In other words, the generator is trained to fool the discriminator. The discriminator is trained like most classification networks are, to maximize the likelihood that its class prediction is correct. This corresponds to a two-player minimax game with the following value function $V(G, D)$:

$\min_G \max_D V(D, G) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$

where G is the generator, D is the discriminator, x is a real image, and z is random noise.

Training each model based on the other's output ideally results in both networks improving each other. As the generator learns to produce more realistic images, the discriminator will have to learn more fine-grained distinctions between *real* and *fake* images. The generator is trained using a signal from the discriminator's prediction via backpropagation, so an improved discriminator will improve the generator during the generator training.

A variant of a generative adversarial network is a conditional generative adversarial network (C-GAN). In a C-GAN, the generator is provided an additional condition as input, and produces output that matches that condition. The discriminator then predicts if the generator output is *real* or *fake* given the condition. For example, an image label could be a condition for a dataset of animal images. The generator would receive a noise vector and a label, such as *dog*, and the discriminator would predict if the image produced is a real *dog* image from the dataset. Given a condition y (such as *dog*), the value function is:

$$\min_G \max_D V(D,G) = E_x[\log(D(x|y))] + E_z[\log(1 - D(G(z|y)))]$$

2.2 Conditional Generative Adversarial Networks for Image Segmentation

Conditional generative adversarial networks have been used for image segmentation [Luc et al. (2016)]. In an image segmentation task, the goal is to accurately classify each pixel in an image into a category. There are two types of image segmentation: semantic and instance. Instance segmentation labels each pixel as belonging to a specific object, where every object in the image has a separate label. Semantic segmentation labels each pixel as a category, where multiple objects in the image of the same type may have the same label. This work focuses on semantic segmentation. Image segmentation is typically evaluated with a mean intersection-over-union score, which is the intersection of the predicted labels and the true labels (the true positives), divided by the union of the predicted labels and the true labels (the true positives plus the false positives plus the false negatives).

In a C-GAN for image segmentation, an image is given as a condition, and the category label of every pixel is predicted by the generator. The set of labels for an image is called the segmentation map. The discriminator then predicts if the segmentation map is real or generated. Luc, et al. were the first group to use a C-GAN for semantic image segmentation [Luc et al. (2016)]. Using a C-GAN for image segmentation produced more accurate image segmentation compared to using a baseline model with the same architecture as the generator. The baseline model was trained with

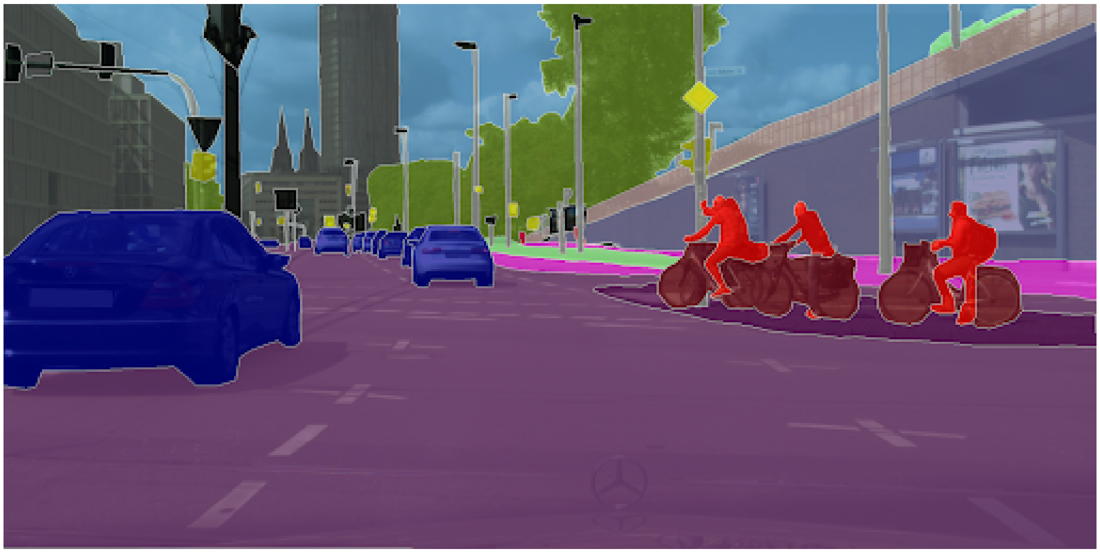

Figure 2.1 One image from the Cityscapes gtFine dataset, with the segmentation labels overlayed.

cross-entropy loss, while the C-GAN model was trained with both the cross-entropy loss and an additional adversarial loss.

Notably, the addition of the adversarial loss helped prevent the model from overfitting the data. During training, the C-GAN had a lower per-class accuracy than the baseline model on the training data, but a higher per-class accuracy on the validation data. This improvement also showed on the test set data, with the C-GAN acheiving higher per-class accuracy, per-pixel accuracy, and mean intersection-over-union scores than the baseline model [Luc et al. (2016)].

2.3 Features Learned by Discriminators

Luc, et al. proposed that adding an adversarial loss was effective for improving image segmentation because it was able to enforce forms of higher-order consistency that could not be enforced or measured by a per-pixel cross-entropy loss [Luc et al. (2016)]. This intuitive explanation makes sense, but the paper did not provide evidence of what exactly the features of higher-order consistency are. Could there have been additional features learned by the discriminator that would not be classified as features of higher-order consistency? Recent research has shown that image classifiers, including the discriminators that are used in a GAN, are able to learn features that are incomprehensible to humans [Ilyas et al. (2019)]. This raises the possibility that an adversarial loss is effective at improving image segmentation because the discriminator learns a variety of features, including features that are incomprehensible to humans.

These incomprehensible features were discovered by investigating methods that an attacker might use to fool an image classification system. One method of attack might be to minimally change some pixel values — so minimally that the change is imperceptible to humans — in an image so that the image classification network will misclassify the image. Examples of these images are called adversarial examples since they are produced by an adversary of the image classification system. (Adversarial examples should not be confused with data produced by a GAN as discussed above. The mechanism to produce adversarial examples is significantly different, but is not the focus of this work).

In *Adversarial Examples Are Not Bugs, They are Features*, Ilyas, et al. claim that adversarial examples represent real features of image classes [Ilyas et al. (2019)]. They term these features as “non-robust features”, and features that are perceptible to humans as “robust features”. They show that the patterns of the imperceptibly perturbed pixels, the non-robust features, are actually highly predictive features of the image data distributions. They show this with two experiments.

For these experiments, “robust accuracy” is defined as accurate classification of a set of images that includes adversarial examples. “Standard accuracy” is defined as accurate classification of images that do not include adversarial examples. In the first experiment, Ilyas, et al. removed non-robust features from the dataset to create a training dataset of only robust features. A classification model trained on this dataset yields good standard accuracy and good robust accuracy on an unmodified test set of images. This indicates that properties of the training dataset effect a classification model’s vulnerability to adversarial examples. In the second experiment, the authors create a training dataset of images that appear to be mislabeled to humans, but in fact have non-robust features associated with their label. A classification model trained on this dataset yields good standard accuracy, indicating that a model can utilize non-robust features effectively.

The demonstration of the existence of non-robust features in image data by Ilyas, et al. is quite surprising, and opens up new lines of inquiry about how to best design networks that learn effectively. Researchers have naturally focused on building architectures to learn robust features from image data, but could different architectures be more optimal for learning non-robust features? It is likely difficult and non-intuitive to design a network that optimally learns features that are imperceptible to humans, since we don't fully know what these features are. However, these features are highly predictive of image classes, and could be utilized to improve image classification accuracy.

2.4 Adversarial Examples fro Image Segmentation

Adversarial examples have also been shown to exist for image segmentation. In *Adversarial Examples for Semantic Segmentation and Object Detection*, Xie, et al. develop an algorithm for effectively creating adversarial examples for image segmentation. Some of the adversarial examples they create are able to cause dramatic errors in the segmentation examples, such as predicting a segmentation map with the pixel label *airplane* predicted in the shape of letters. These examples were shown using the Fully Convolutional Network (FCN) model [Xie et al. (2017)].

Adversarial examples being successful for image segmentation is somewhat more surprising than adversarial examples being successful for image classification, since image segmentation requires a prediction for each pixel rather than a single class prediction. Additionally, the information captured by the FCN model to predict each pixel is heavily tied to the corresponding pixel in the same location in the input image. This led me to hypothesize that the erroneous pixel class predictions are most heavily influenced by the pixels in the same location in the input image, and that these pixels likely contain learnable features.

2.5 Features of Image Segmentation Errors

It is possible that some non-robust features could be more easily learned from specific augmentations of the input data than from the unaltered input data. I hypothesized that the specific pixels that were misclassified, along with the incorrectly predicted labels, would collectively contain learnable features. Further, I hypothesized that these features could be learned with a second discriminator. If a discriminator can learn non-robust features in images, then a discriminator might also be able to learn non-robust features in this data representation.

In order for the generator to benefit from a second adversarial loss, the loss needs to be able to be backpropagated. This requires the input to the second discriminator, the segmentation errors, to be created in a way that is differentiable. One way to do that is by predicting the segmentation errors with a second generator.

2.6 Stacked Conditional Generative Adversarial Networks

Using multiple GANs to generate multiple predictions from an image has been used before, in the context of shadow detection and removal. In *Stacked Conditional Generative Adversarial Networks for Jointly Learning Shadow Detection and Shadow Removal*, Wang, et al. designed an architecture that uses two generators and two discriminators. The first generator takes an image as input and predicts the shadow mask of the image. The first discriminator takes both the original image and a shadow mask (either predicted or real) as input, and classifies the input as either real or fake. The second generator takes both the original image and the predicted shadow mask as input, and predicts a shadow-free image. The second discriminator finally takes the original image, the shadow mask, and the shadow-free image as input and predicts whether the input is real or fake. The shadow mask and shadow-free image will either both be fake or both be real when they are used as input for the second discriminator [Wang et al. (2018)].

I use this same architecture, substituting in segmentation label predictions instead of shadow mask prediction, and segmentation error prediction instead of shadow-free image prediction. The first generator predicts a segmentation map, and the second generator predicts the errors in the predicted segmentation map. Specific details about this architecture are given in the *Methods and Procedures* section. Aside from changing the prediction tasks of both generators, I also make a conceptual change to the input of the second discriminator. The second discriminator is always given as input the original image, a *predicted* segmentation map, and *real or predicted* segmentation errors. This design choice was made to ensure that the second discriminator does not simply learn to detect when there are no segmentation errors, since the segmentation errors calculated from a real segmentation map would always be a tensor of zeros.

A major benefit of the stacked GAN architecture is that both prediction tasks enhance each other. Wang, et al. demonstrate that predicting the shadow mask and then the shadow-free image is able to achieve better predictions of both compared to a typical multi-task learning setup, where the image is transformed into a shared embedding that both the shadow mask and shadow-free image are predicted from. This is because the stacked GAN architecture allows information to flow from the first task to the second during the forward pass of the network, and from the second task to the first during backpropagation of the loss function [Wang et al. (2018)]. For image segmentation, the information flow from the second task back to the first is the primary benefit of using this architecture. This allows any features learned by the second discriminator to effect the first generator via backpropagation.

CHAPTER 3. METHODS AND PROCEDURES

3.1 Data

Since this experiment relies on image segmentations errors, the accuracy of the segmentation labels is very important. The Cityscapes dataset [Cordts et al. (2016)] contains a subset of images that are finely labeled, such as the example in Figure 2.1. These 5000 finely labeled images were loaded using the Cityscapes class from PyTorch Datasets. Each image and segmentation map was resized from 1024x2048 to 256x512. Each pixel value of the images was normalized to be a value between zero and one by dividing it by 255. The segmentation labels were converted into one-hot tensors, with the training ID of training classes 0-18 corresponding to indices 0-18 in the third dimension, and every other training ID being assigned to index 19. The intersection-over-union score used for evaluating the models only took into account indices 0-18, consistent with the Cityscapes benchmark evaluation [Cordts et al. (2016)].

3.2 Model Architecture

The model architecture was directly inspired by the architecture specified in *Generative Stacked Conditional Generative Adversarial Networks for Jointly Learning Shadow Detection and Shadow Removal*. Changes were made to number of input and output channels, and the final activation function on the first generator was switched to softmax.

The ground truth representation of segmentation errors was calculated by converting the predicted segmentation map tensor to a one-hot tensor corresponding to the maximum predicted values, and then subtracting this predicted segmentation map tensor from the ground truth segmentation map one-hot tensor. This leaves errors in the form both 1s and -1s, corresponding to the correct labels and the incorrect predictions. The tanh activation function on the last layer of the second generator corresponds to this [-1, 1] range.

Table 3.1 The architecture for both generators, with a / corresponding to first/second where needed. Cv indicates a classic convolutional layer and CvT indicates a transposed convolutional layer that upsamples a feature map. LReLU is short for Leaky ReLU activation and BN is an abbreviation of Batch Normalization. The *link* row shows when the output of one layer is concatenated to the input of another layer, consistent with the standard U-Net architecture.

Layer	Cv $_0$	Cv $_1$	Cv $_2$	Cv $_3$	Cv $_4$ (×3)	Cv $_5$	CvT $_6$	CvT $_7$ (×3)	CvT $_8$	CvT $_9$	CvT $_{10}$	CvT $_{11}$
Channels In	3/23	64	128	256	512	512	512	1024	1024	512	256	128
Channels Out	64	128	256	512	512	512	512	512	256	128	64	20
before	–	LReLU	LReLU	LReLU	LReLU	LReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
after	–	BN	BN	BN	BN	–	BN	BN	BN	BN	BN	Softmax/Tanh
link	→ CvT $_{11}$	→ CvT $_{10}$	→ CvT $_9$	→ CvT $_8$	→ CvT $_7$	–	–	Cv $_4$ →	Cv $_3$ →	Cv $_2$ →	Cv $_1$ →	Cv $_0$ →

Table 3.2 The architecture for both discriminators, with the same abbreviations as Table 3.1.

Layer	Cv $_0$	Cv $_1$	Cv $_2$	Cv $_3$	Cv $_4$
Channels In	23/43	64	128	256	512
Channels Out	64	128	256	512	1
before	–	LReLU	LReLU	LReLU	LReLU
after	–	BN	BN	BN	Sigmoid

3.3 Loss Functions

The generators of this network were trained with both a data loss function and an adversarial loss function, while the discriminators were trained only with binary cross-entropy loss. The data loss function for the first generator is cross-entropy loss, while the data loss function for the second generator is L1 loss. These loss functions together can be expressed as playing a two-team minimax game, where the generators are one team and the discriminators are the other team, with the value function:

$$\min_{G_1,G_2} \max_{D_1,D_2} CE\left(G_1\right) + \lambda_1 L1\left(G_2|G_1\right) + \lambda_2 L_{CGAN_1}\left(G_1, D_1\right) + \lambda_3 L_{CGAN_2}\left(G_2, D_2|G_1\right)$$

where λs are hyperparameters. All λs were 1 in this experiment.

The discriminators loss functions were both implemented using PyTorch's binary cross-entropy loss, BCELoss, where the discriminator output and a tensor of equal size containing 0s or 1s were passed as input. The first generator's loss function was implemented with PyTorch's negative log likelihood loss, NLLLoss, where the log of the first generator's output and a tensor containing class labels at each pixel location were passed as input. Note that cross-entropy loss is equivalent to the combination of log(softmax) and negative log likelihood loss. The second generator's loss function was implemented with Pytorch's L1 loss function, L1Loss, where the second generator's output and the calculated ground truth errors were passed as inputs.

3.4 Training

During training, either the discriminators' parameters or the generators' parameters were updated at the end of each epoch. Alternating which parameters are updated every epoch was the best found alternating frequency. Only the first fifth of the Cityscapes dataset was used for training in order to train networks more rapidly on limited computational resources. The performance of models would likely increase when trained on the full dataset, while their relative performance compared to each other would likely remain the same.

CHAPTER 4. RESULTS

This experiment aimed to show that the addition of a second adversarial loss via a second GAN stacked on top of the first GAN would improve the performance of the first generator. The performance of the first generator was measured with an mean intersection-over-union score of the 19 classes recorded in the Cityscapes benchmark. The additional adversarial loss did not significantly effect the performance of the first generator during training.

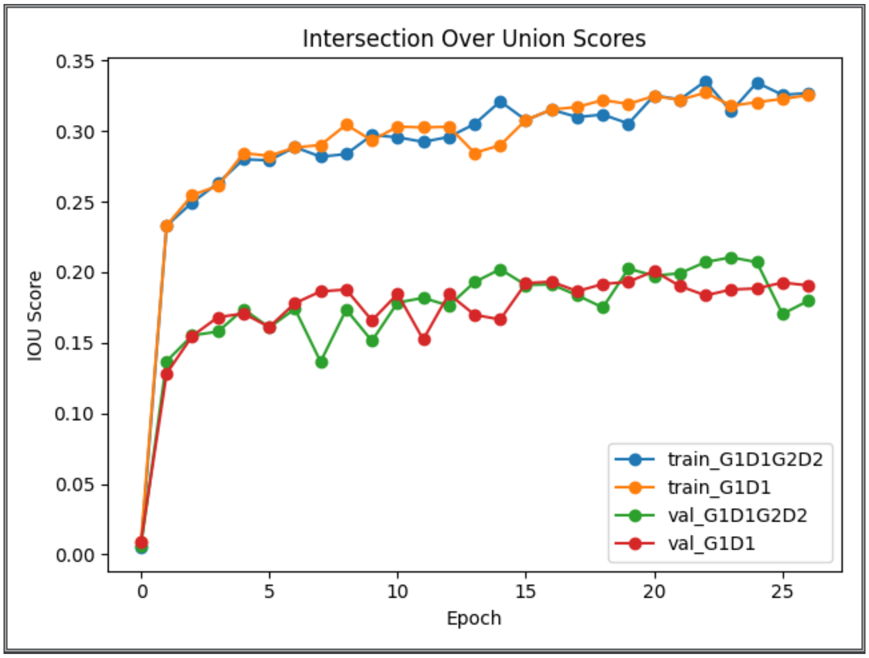

Figure 4.1 The mean IoU scores of each epoch during training of a single GAN, labeled G1D1, and the mean IoU scores of each epoch during training of a stacked GAN, labeled G1D1G2D2. Scores on both the training set and the validation set are shown.

However, the mean intersection-over-union scores of these models did significantly differ when evaluated on the test set.

Table 4.1 Cityscapes Test Set Mean Intersection-Over-Union Scores

Model	**Intersection-over-union**
Single GAN	0.370
Stacked GAN	0.498

Interestingly, the intersection-over-union scores on the test set were actually higher than the scores on the training set and the validation set. Typically, we would expect that the test set scores would be about the same as the validation set scores, or possibly a bit lower, with the validation set scores being a little lower than the training set scores. However, this result can be explained by some layers of the network behaving differently during inference mode than they do during training mode.

Batch normalization layers calculate the mean and variance that are used to normalize the output in different ways depending on if the network is in training mode or inference mode. In training mode, the mean and variance are estimated from the current batch that is being processed. A running estimate of the global mean and variance is updated during training, but not used to process each batch. These global estimates are used during inference mode. Which mean and variance statistics are used within these layers can dramatically effect the overall performance of the network.

If the batch size is too small, the estimates of the mean and variance during training can be inaccurate. If the training is also unstable, meaning that the statistics of activations change rapidly during training, the running estimates of the global mean and variance can also be very inaccurate. GAN networks are notoriously unstable to train, and were constrained to small batch sizes for this work. This may have diminished the performance of both networks during training and during evaluation on the validation set.

When the models were reloaded from saved files for inference on the test set, the mean and variance used in batch normalization layers were initialized to default values. Using these default values for normalization effects the performance of both models during inference, which explains how the networks were both able to achieve higher intersection-over-union scores on the test set.

CHAPTER 5. SUMMARY AND DISCUSSION

This experiment aimed to learn additional features of images that had not previously been shown to exist, and to utilize those features to improve image segmentation. After investigating adversarial examples for image segmentation, I hypothesized that the features which caused pixel miss-classifications are locally tied to the pixels which are miss-classified. I further hypothesized that these features could be learned by providing the images and indications of which pixels were miss-classified as input to a discriminator. I referred to these inputs as image segmentation errors.

To determine if these features could be learned and leveraged to improve image segmentation, I compared the performance of a single GAN for image segmentation to the performance of a stacked GAN for image segmentation. The second GAN of the stacked GAN was designed to learn these features. This comparison was made using the Cityscapes dataset.

The results of this performance comparison was inconsistent and did not conclusively provide evidence that the errors of segmentation predictions contain features that can be learned and leveraged to improve image segmentation. During training, the stacked GAN showed no improvement over the single GAN. However, the stacked GAN did show improvement on the test set. It is unclear whether this result was caused by the additional adversarial loss helping the stacked GAN generalize more effectively to unseen data, or if this result was due to chance. Resetting certain variables in batch normalization layers before evaluation on the test set improved the performance of both networks, and more data needs to be collected to determine if these variables caused a discrepancy between the single GAN performance and the stacked GAN performance. Further experiments could be conducted to clarify these results.

5.1 Future Work

Future experiments could resolve any ambiguity caused by unstable training with batch normalization layers. This can be done by increasing the batch size during training, which may require larger computational resources than were available for this experiment. However, using gradient accumulation techniques can allow for increasing the batch size with only a minor increase in memory requirements. Additionally, batch normalization layers could be eliminated entirely. One potential challenge with these experiments is maintaining stable training of GANs. This was one of the major challenges with the current work. Many training variables, particularly the frequency with which to alternate updating generators or discriminators, need to be tuned in order for the training of both networks to converge properly.

Further experiments could also be conducted to determine the effect that the architecture of the generator has on which segmentation errors are produced. It is possible that residual connections or similar connections preserve the locality of the features that cause errors to the pixels that are miss-classified. An architecture without these connections in the generator may have varying success at learning features from the segmentation errors produced by the generator.

This work could also be applied to different types of data. There was specific motivation for attempting to learn features from segmentation errors, but it may be possible to learn features from a variety of types of errors. Data that can be formatted similarly to images would be particularly amenable for applying the same type of stacked GAN architecture, but additional architectures could be designed to learn the features of errors of diverse data types such as text data or graphical data. Other ongoing research investigating the types of features that known architectures learn may help inform decisions about how to design these additional architectures.

REFERENCES

Cammarata, N., Carter, S., Goh, G., Olah, C., Petrov, M., and Schubert, L. (2020). Thread: Circuits. *Distill*, 5(3):e24.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410.

Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*.

Wang, J., Li, X., and Yang, J. (2018). Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1788–1797.

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., and Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378.