Feature To Adapt

Boosting accuracy of UDA for pixel-level semantic segmentation by targeting direct Feature Space alignment using L2 norm loss

Omer Landau
Tel-Aviv University

Abstract

Training deep neural models for pixel-level semantic segmentation is a difficult task which usually requires a large number of labeled samples. In turn, these types of datasets are expansive to create and needed for variant real life applications. Moreover, while supervised approaches based on convolutional neural networks constantly break new records (on old benchmarks). In most of real life applications there is a big gap between data distribution of train and test domains, causing these models to degrade on realtime environments. In order to reduce the cost of labelling and also improve the generalisability of existing models, unsupervised domain adaptation approaches were proposed. Using unlimited knowledge coming from synthesised labeled data, these models are trained on synthetic images and labels and generalise to work on real world target data in an unsupervised fashion. Till this day, most of the leading UDA setups for semantic segmentation, were focused mainly on adapting through output space and neglecting direct influence and use of lower layers features. In this work, we have demonstrated that this kind of influence is helpful, boosts results, and can be achieved by regulating backbone's features to have larger L_2 norm. We've shown that larger norm feature extraction Is a good practice for boosting overall adaptation results and may even be used as an adaptation method on its own. Providing fair and competitive results with light computationally addition to existing supervised semantic segmentation models. Finally we achieve SOTA results on GTA5 to Cityscapes domain adaptation for pixel-level segmentation task, using the latter boosting method on existing architectures. Our code along with documentation is available at github.

1. Problem Settings

As mentioned before We focus on Unsupervised Domain Adaptation (UDA) for semantic segmentation.

For this type of problem we have access to the source data X_S with its pixel-level labels Y_S and also access to the target data X_T without any target labels at all. Hence, given x_s, y_s and x_t drawn from the above sets, the goal of unsupervised domain adaptation is to transfer knowledge from the labeled source set to the unlabelled target set. More specifically our target is to learn an adequate model G which can accurately predict semantic pixel-level segmentation for any $x_t \in X_T$. The generative model G is almost identical in all of our setups, so as the segmentation loss, which is unchanged among different architectures. The entire problem can be formalised as following, firstly the model G is set to minimise fully supervised segmentation loss on the source domain: $L_{seg}(G) = E[l(G(X_S), Y_S)]$ where $E[]$ denotes statistical expectation and l is the appropriate loss - In our case multi-class cross entropy log loss.

Secondly when using adversarial setup, UDA methods train G to acquire domain invariant features by attacking a domain discriminator D. The discriminator in its turn is trained to distinguish between the segmentation outputs of target and source domains. This can be accomplished by minimising and maximising the following adversarial loss:

$$L_{adv}(G, D) = -E[log(D(G(X_S))] - E[log(1 - D(G(X_T))]$$

As mentioned above D is trained on the output tensor of G, meaning the pixel-level segmentation that G predicts. Usually in adversarial domain adaptation, models are focusing on direct feature space alignment rather than output space. In the case of pixel-level segmentation, this is not true. Most of the latest leading adversarial settings for Semantic segmentation UDA, are focusing mainly on output space alignment, calculating D's loss on $G's$ prediction. That way they are almost totally neglecting direct influence on lower "backbone" features of G. As we will show later, the latter leads to performance degradation which can be fixed. This leads us to the final component, which isn't necessarily part of the original problem setting, but is used in all of our setups. That is a step wise L_2 norm loss (described later):

$L_{norm} = E[l_{step}(F(X_S))] + E[l_{step}(F(X_T))]$ where $F \subset G$ is a function created by subset of parameters from G. In our neural network architectures F would be considered as G's backbone.

Due to limited sources and time, we chose to focus on a specific task in the problem domain of UDA for semantic pixel-level segmentation, our chosen benchmark is adapting GTA5 to Cityscapes.

Datasets

Cityscapes is a real-world dataset containing 5,000 annotated images of street scenes with 2048×1024 resolution, taken from streets in Germany. We use Cityscapes as our target domain (X_T, Y_T). GTA5 is a dataset synthesised from the GTA5 game, containing 24,966 annotated images with 1914×1052 resolution. We use GTA5 as our source domain (X_S, Y_S). For our classes, we use 19 common labels of GTA5 and Cityscapes tested by all of previous works. The classes are: road, sidewalk, building, wall, fence, pole, light, sign, vegetation, terrain, sky, person, rider, car, truck, bus, train, motorcycle and bicycle.

2. Method

Direct Feature Space Adaptation

While in most UDA problems, models are trying to directly align feature space of both domains. It is not true for UDA semantic segmentation task. Since Tsai et al. [1] presented an output space adaptation theorem, all of the latest leading efforts in the field, have mainly focused their adaptation methods (whether if it's adversarial or not) on target and source outputs coming from the very high-end of the architecture - the classifiers. Different methods, such as SWD [5], CLAN [2] or MaxSquare [3] are all training an objective function which is focusing mainly on predicted data - no features. There are plenty good reasons for using this strategy. Pixel-level segmentation is quite a delicate task and most of former adversarial method which have tried to rely on direct feature space alignment such as CyCADA [7] accomplished quite low results comparing to the current standard.

Having said that, in the domain of classification UDA problems, plenty of models are focusing on direct feature space alignment. Even more so, Xu et al. [4] showed that by simply adding a loss which encourages classifier's backbone features to have large L_2 norm on both source and target images, they can accomplish adaptation and transfer knowledge from source to target domain. They've tested this theory on simple classification tasks such as VISD17 challenge or Office-Home presenting adequate results.

Searching for a way to make direct Feature Space alignment relevant and helpful to our task. We have come to think that the latter types of L_2 norm objection functions might be also useful for semantic segmentation. More specifically we've chosen to focus on Step Wise Adaptive Feature Norm [4], and rescaled it to work in our different delicate conditions. The Step Wise Adaptive Feature Norm works as following:

$$L_{norm} = \frac{\lambda}{n_s + n_t} \sum_{x_i \in X_S, X_T} L_d(h(F(x, \theta_0)) + \Delta r, h(F(x, \theta)))$$

Where **F** is a feature extractor backbone, θ_0 is the previous updated model parameters, θ is the updating current parameters. λ is a scalar, n_s, n_t are the number of images in each source and target sets. Δr is a scalar representing the increment that we encourage in the feature's L_2 norm at each step. h and L_d are L_2 norm and distance calculation which will be explained later on in the implementation section. Step wise strategy is chosen since this way we can encourage backbone features to slightly increase at each step, in a stable and controlled way without pointing all of the features into a simple certain fixed direction. Each feature is encouraged to increment its L_2 norm, but the incrementation is **relatively** to its previous measure, that way the absolute growth is also diverging between different features, allowing the features to preserve its unique identities.

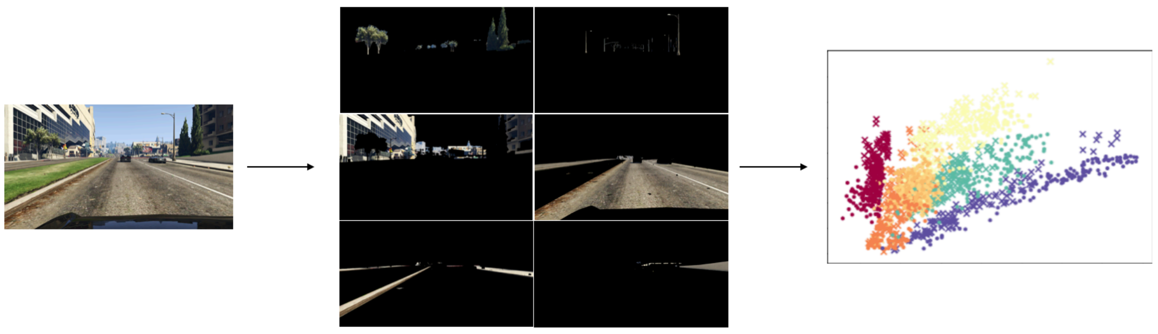

Figure 1. The above image presents a process of splitting down to classes images from both source and target domain. Then dimensionally reduct it and extract features from it using PCA. PCA is used since we are interested in the influence of L_2 norm on the features, and while PCA persevere distances in a certain manner, it is not guaranteed when using other dim reduction method such as T-SNE. Finally we can see that indeed for features with larger L_2 norm - 'x' which stands for GTA5 images is more aligned

with '.' which stands for Cityscapes images. This strengthens our intuition that adding L_2 norm objection function might help adaptation of feature space, as we go forward with this task.

Strategy

First thing, this work isn't about presenting an entirely new architecture. Rather it addresses the issue of direct influence on feature space layers (or the lack of it), and offers a method to fix it for boosting overall performance of existing models. Having said that, In order to test our method, we pick two leading architectures. One is adversarial (CLAN [2]) and the other is not (MaxSquareLoss [3]). Both are focusing mainly on output space adaptation. We show that by combining these methods with an additional component which addresses feature space alignment, better adaptation results can be achieved. We also train a third model, which is adapted only by Feature Space - meaning it is trained using only segmentation loss on source predictions and Step Wise Norm Loss on target and source feature extractions.

We come up with the third model for two main reasons:

1. It acts as an additional baseline for showing the effect of feature space alignment - in an obvious way since we are training only using feature space alignment (achieved by Step Wise L_2 norm loss). Indeed the results are quite impressive and the outcome model performs even better than the baseline adversarial architecture.
2. So one can use this model to boost performances in an easy way, without the need of retraining an entire existing architecture. We show that the Feature Space knowledge produced by this "Only Norm" trained model, can be successfully transferred even by using simple ensemble - weighting the outputs from existing models and the "Only Norm" model. We demonstrate this method of knowledge transfer on both CLAN original model, and MaxSquareLoss original model.

Due to lack of time and resources, we go through training process from the beginning only in the Adversarial case and of-course for "Only Norm" architecture. We demonstrate the boosting of MaxSquareLoss [3] model, only by using ensemble of our Norm Only model along with MaxSquareLoss [3] Pre-trained model. providing another evidence for successful knowledge transfer from Feature Space in the light way.

Implementation Details

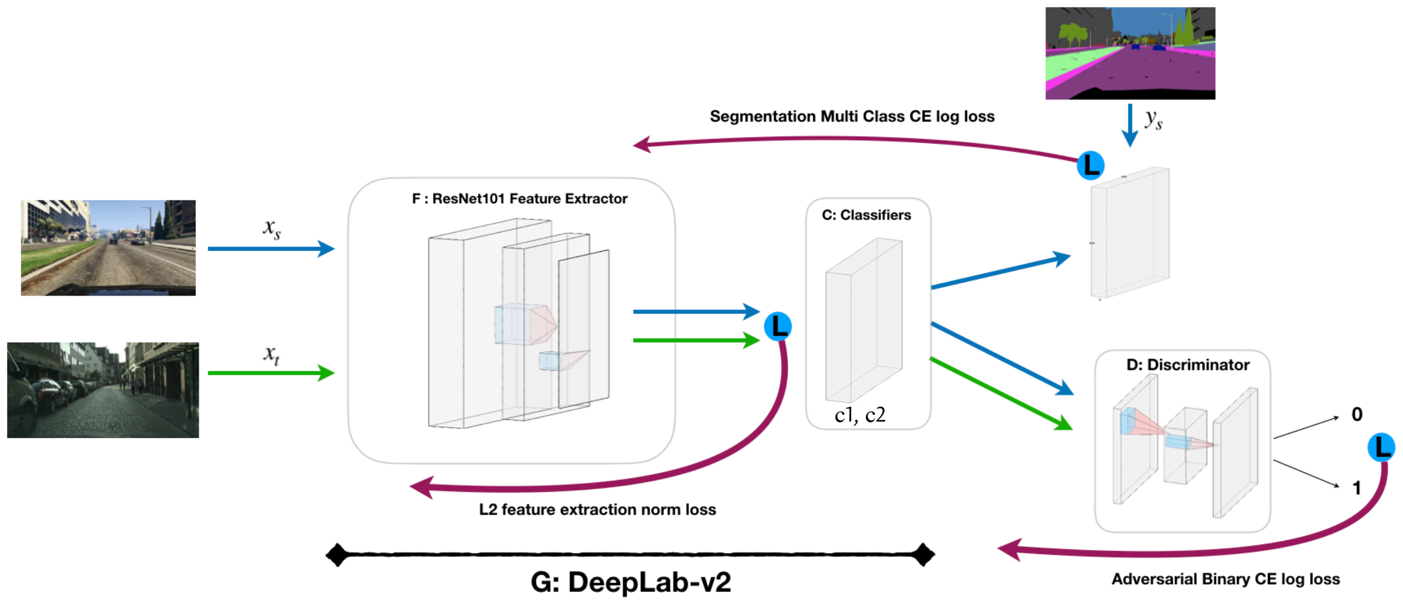
Segmentation Multi Class CE log loss
y_s
L
x_s
x_t
F : ResNet101 Feature Extractor
C: Classifiers
c1, c2
L
D: Discriminator
0
1
L
L2 feature extraction norm loss
G: DeepLab-v2
Adversarial Binary CE log loss

Figure 2. Network architecture, schematic draw of most of our model components. Consists feature extractor F, our classifiers as C and the discriminator for the adversarial setup. An additional component which is not mentioned above, is coming from the use of CLAN [2] as a baseline architecture. That is a discrepancy loss - which is applied on the weight's tensor of the classifiers. It is a simple cosine similarity measure which encourages the two classifiers do differ from each other. Practice results shows that the two classifies are almost identical despite the use of this component.

Segmentation Network

Used by majority of previous works on the benchmark, we base our segmentation framework on AdaptSegNet's (Tsai et al. [1]) segmentation network. The segmentation network is basically DeepLab-v2 with ResNet-101 model pre-trained on ImageNet as backbone. Due to memory issues, multi scale fusion isn't used. The last classification layer of the model is removed, and the two last convolution layers are modified - changing their strides from 2 to 1 - resulting the feature tensors to be 1/8 times the input image size. In order to enlarge receptive field, two dilated convolution layers are applied (conv4 and conv5) with a stride of 2 and 4 respectively. For the classifiers, Atrous Spatial Pyramid Pooling (ASPP) is used. The final predicted segmentation output is of shape (19, 65, 129) so an additional Upsample layer is applied in order to interpolate it to 2048×1024 which is Cityscapes's images resolution. The interpolation method is bilinear. When trained in a supervised manner on Ciryscapes, the segmentation network scores 65.1 mIoU. So our "oracle" results should be around 65.1 mIoU.

Discriminator

In the adversarial setup, we also use a discriminator which was also introduced by Tsai et al. [1]. The network consists 5 convolution layers with 4×4 kernel and stride of 2, the channel number is

{64, 128, 256, 512, 1}. Each convolution layer is followed by leaky ReLU activation, except for the last one. The final layer is an upsample layer for re-scaling the output to the size of the input.

Objective function

Segmentation loss

$$L_{seg}(G) = \sum_{x_s \in X_S, y_s \in Y_S} \sum_{i=1}^{129\times65} \sum_{c=1}^{19} -y_{s,i,c} log(G(x_s)_{i,c})$$

It's a simple multi class CE log loss. Where x_s, y_s are images and labels, 129×65 is the resolution of predictions of **G**, 19 is the number of classes. And $G(x_s)$ is a prediction of x_s through **G**.

Adversarial loss (only in adversarial setup)

$L_{adv}(G, D) = -E[log(D(G(X_S)))] - E[(\lambda_{local}M(p_1, p_2) + \epsilon)log(1 - D(G(X_T)))]$ Where p_1, p_2 are **G**'s predictions made by both of the classifier respectably. M denotes a cosine distance between the predictions, element wise: $1 - cos(p_{1,i,j}, p_{2,i,j})$. This was applied in CLAN's [2] original objective, in order to emphasise differences between variant category-level outputs.

Discrepancy (only in adversarial setup)

The discrepancy loss is one of two main addition that CLAN [2] has made to the former Adversarial baseline (AdaptSegNet [1]). It is applied on the weight tensor of the classifiers and is a simple cosine similarity measure. The purpose is to encourage the classifiers's weights do differ from each other. It is worth mentioning that **Practice results show that the two classifiers are almost identical despite this component**.

$$L_{weight}(G) = \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{||\overrightarrow{w_1}||||\overrightarrow{w_2}||}$$

Step wise norm loss

$L_{norm}(F(x, \theta_0), F(x, \theta)) = (||F(x, \theta_0)||_2 + \Delta r - ||F(x, \theta)||_2)^2$ where **F** is **G**'s backbone, and θ_0 symbols the weights of previous round (not updated) and θ symbols the current weight which we update. Δr is an unchanged scalar. The above power is element wise, also the final loss is the mean of the above, in order to achieve a scalar output for the loss. We use this objective on both images from source and target domains, meaning: $x \in X_T \cup X_S$.

Final objective: $arg \min_G \max_D L(G, D)$, $L = \lambda_{seg}L_{seg} + \lambda_{norm}L_{norm} + \lambda_{disc}L_{weight} + \lambda_{adv}L_{adv}$
We solve this by optimising G and D in turns, explained in the training procedure below.

Training Procedure

We first freeze **D**'s grads and feed forward source image x_s to **G** to generate the feature extractor: $F(x_s)$ and two predictions $p_{s,1} = C_1(F(x_s)), p_{s,2} = C_2(F(x_s))$. Then we calculate backwards norm loss $l_{norm}(F(x_s))$ and right after that we calculate backwards segmentation losses: $l_{seg}(p_{s,1}, y_s) + l_{seg}(p_{s,2}, y_s)$. Now we feed forward target image x_t to **G** and generate the feature extractor: $F(x_t)$ and two predictions $p_{t,1} = C_1(F(x_t)), p_{t,2} = C_2(F(x_t))$. Then we calculate backwards norm loss $l_{norm}(F(x_t))$ and right after that we calculate adversarial loss, with the predictions $p_{t,1} + p_{t,2}$. Now instead of label $p_{t,1} + p_{t,2}$ as a target prediction, we label it as source prediction - this is done in-order to train **G** to trick **D** into guessing that target samples are actually source ones, and by that maximise **D**'s loss. So the adversarial loss looks as following: $l_{adv}(softmax(p_{t,1} + p_{t,2}), \ source \ label)$. After we finish with the adversarial loss we calculate the Discrepancy loss $l_{weight}(w_{c_1}, w_{c_2})$. Now it's time to unfreeze **D**'s weights, and train **D**. Once again, we feed forward x_s, x_t through **G** - acquire 4 predictions $(p_{s,1} + p_{s,2}, p_{t,1} + p_{t,2})$ and feed each of them forward to **D** and calculating $l_d(softmax(p_{t,1} + p_{t,2}), target \ label)$. We go through the above procedure for each training batch. For training in the non-adversarial setup (Norm only model), we just skip the adversarial losses, the discrepancy loss and do not use the network **D**. Everything else stays the same.

We train our model on Titan X GPU with 12 GB memory. Using a batch size of 1 for a maximum of 100,000 steps and crop our inputs to 512×1024. We use SGD as the optimiser for **G** with momentum of 0.9, and Adam as the optimiser for **D** with $\beta_1 = 0.9, \beta_2 = 0.99$. Both optimisers are set to weight decay of $5e - 4$. For SGD, we initiate the learning rate with a value of $2.5e - 4$ and set a poly learning rate policy with power of 0.9 for decaying. Adam is initialised with $5e - 5$ learning rate. Instead of early stopping we use damping for both adversarial and norm losses.

<u>We set our hyper parameters as following:</u>

Norm loss - $\lambda_{norm} = 0.00015, \Delta r = 0.12$ - we had to alternate the effect of the norm to be much smaller then previous studies. The main reason for that is due to the fact that pixel-level segmentation is a delicate task. We've come to learn that rapid enlargement in Feature Space norm might help adaptation - but it is also completely ruins the ability of the classifier to reconstruct an adequate segmentation out of the extracted features. So much slower and delicate enlargement process was applied.
Adversarial and Weight losses - $\lambda_{weight} = 0.01, \lambda_{adv} = 0.001, \lambda_{local} = 40, \epsilon = 0.4$. Unchanged from baseline adversarial architecture.

3. Results

Feature space adaptation

In order to validate our theory and the effect of the norm loss on feature space adaptation. We pick 130 images from Val group of both datasets. Each of the images is divided into 19 different pictures, each contains only objects from one class - using the same procedure shown in Figure 1. We then feed forward these class-unique images into our trained model and produce a feature extraction tensor provided by model's backbone (ResNet101 Layer4's output). We further embed this tensor to R^2 using PCA and T-SNE so we could plot and study the results. We go through this procedure with the following trained models:

1. Non adapted - feature extraction from original pictures using only PCA and T-SNE (no feeding through any model)
2. Adversarial only (based on CLAN [2] architecture).
3. Only norm model.
4. Adversarial + Norm model (Adversarial based on CLAN [2] architecture).

The results of this experiment are quite interesting, and show that models which had Norm Loss activated directly on backbones features were adapting feature space **better**.

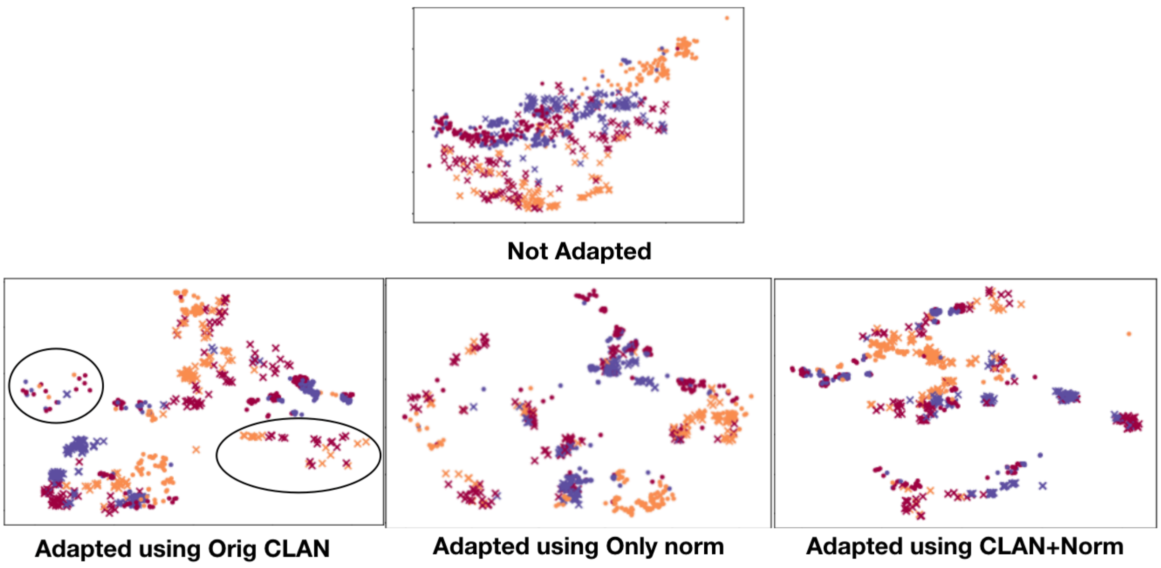

Figure 3. Feature Space Adaptation, scatter after embedding using PCA and T-SNE. Upper image: Non - Adapted (only dim reduction). Lower images left to right: 1. Only Adversarial (Original CLAN). 2. Only norm 3. CLAN + Norm. Clearly one can see that models with norm loss produce significantly more aligned feature space of the two domains. 'x' is for GTA5 features and '.' is for Cityscapes. We plot only 3 random chosen classes for visual reasons (signs, roads and buildings).

But not only that feature space is better adapted, the final predictions on test set are showing an impressive increment in mIoU (mean intersection over union) of the the generated predictions compared to the Ground Truth segmentations. Meaning that the improvement in feature space adaptation contributed to knowledge transfer between the domains and also improved the segmentation quality of output space predictions.

Final IoU results

We further validate our model alteration, using the standard metric for pixel-level segmentation IoU - intersection over union, sometimes called jaccarad index. The formula of the metric is simple and goes as following:

$\forall i \in L_T: \quad J(p_{t,i}, y_{t,i}) = \dfrac{|p_{t,i} \cap y_{t,i}|}{|p_{t,i} \cup y_{t,i}|}$ where $p_{t,i}$ are the pixels of predicted class i for a specific image. and $y_{t,i}$ are the ground truth pixel labels of the class for the same specific image. Can be described as $\dfrac{area\ of\ overleap}{area\ of\ union}$.

We calculate IoU on each of the classes apart and also present the mean IoU result of all classes (referred to as mIoU). The results are as following:

Method	Road	Sdwk	bldng	wall	Fence	Pole	Light	Sign	Vgttn	Trrn	Sky	Person	Rider	Car	Truck	Bus	Train	Mcycl	Bcycl	mIoU
AdaptSeg Net [1]	86.5	36.0	79.9	23.4	23.3	23.9	35.2	14.8	83.4	33.3	75.6	58.5	27.6	73.7	32.5	35.4	3.9	30.1	28.1	42.4
CLAN [2]	87.0	27.1	79.6	27.3	23.3	28.3	35.5	24.2	83.6	27.4	74.2	58.6	28.0	76.2	33.1	36.7	**6.7**	31.9	31.4	43.2
SWD [5]	**92.0**	**46.4**	**82.4**	24.8	24.0	**35.1**	33.4	**34.2**	83.6	30.4	**80.9**	56.9	21.9	82.0	24.4	28.7	6.1	25.0	33.6	44.5
Max Square [3] **	89.4	43.0	82.1	30.5	21.3	30.3	34.7	24.0	**85.3**	39.4	78.2	**63.0**	22.9	84.6	36.4	43.0	5.5	34.7	33.5	46.4
Source only	75.8	16.8	77.2	12.5	21.0	25.5	30.1	20.1	81.3	24.6	70.3	53.8	26.4	49.9	17.2	25.9	6.5	25.3	36.0	36.6
Only Norm loss	86.59	26.65	79.63	27.15	22.61	29.37	37.19	27.39	83.21	31.44	73.7	60.36	**28.91**	81.92	33.41	36.53	1.85	32.23	36.44	44.03
CLAN + Norm *	87.73	25.41	80.32	28.65	23.41	29.47	37.49	26.6	83.93	35.38	74.38	60.39	28.9	83.48	35.42	39.36	1.1	**32.83**	36.27	44.76
CLAN + Norm	88.83	33.17	81.48	32.66	**24.24**	32.28	**37.65**	27.53	83.94	33.88	75.42	60.99	28.11	**84.73**	33.36	42.76	0.97	31.13	36.66	45.67
Max Square + Norm *	89.57	40.32	82.06	**33.58**	23.35	29.77	36.11	23.68	84.33	**39.61**	74.89	60.31	27.3	83.84	**42.8**	**45.95**	1.9	32.16	**37.87**	**46.81**

Table 1. IoU table, the grey cells means that these architectures are ours (former + norm alteration). If the method is marked by * it means that both components were not trained together from beginning but rather it is an ensemble of pre-trained original model and our model trained only using norm loss. We've succeeded to improve both MaxSquare [3] and CLAN [2] architectures, achieving SOTA mIoU results. We've also achieved best IoU for most of the classes. **All of the above results were achieved using the same backbone and segmentation architecture - ResNet101 and DeepLab-v2 .**

Meaning of ** : When we've evaluated the results of the original MaxSquare model ourselves, we've come to discover that the reported results are higher than our outcome - we've observed 44.6 mIoU, and the baseline article reported 46.4 mIoU. We have checked MaxSquare's original evaluation and mIoU codes, they are different than all previous SOTA baselines for evaluation. It is not addressed in the MaxSquare article. All other results were calculated using the same evaluation code (provided by AdaptSegNet [1]). Also different from other methods MaxSquare is using DeepLab-V2 model which is pertained on GTA5 source before adaptation training.

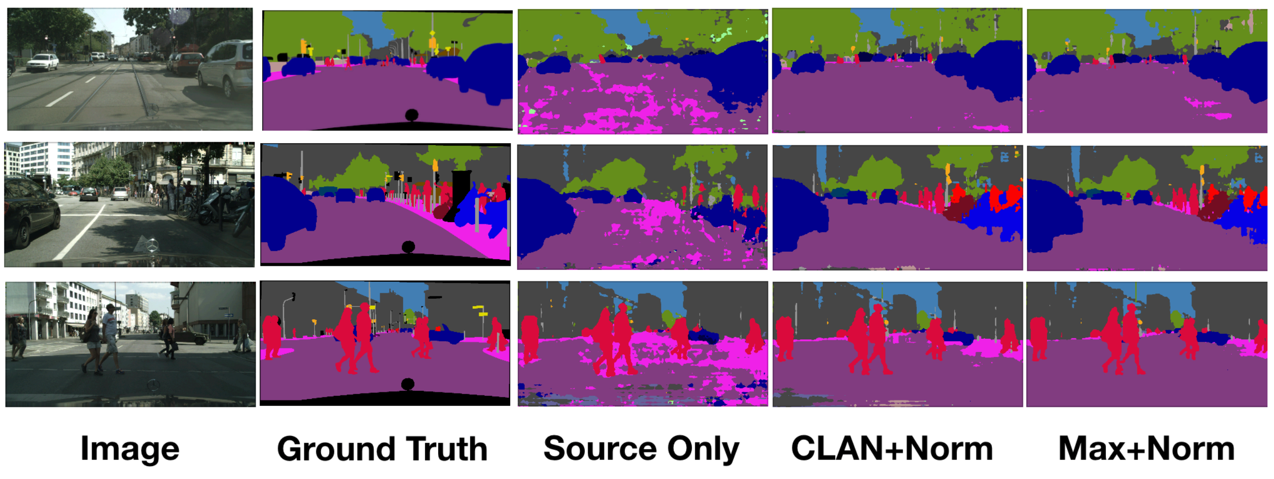

Figure 4. Prediction Visualisation, for each target image, we show the ground truth label, source only (non-adapted model), CLAN+ Norm prediction and Max+ Norm prediction.

Method	Road	Sdwk	bldng	wall	Fence	Pole	Light	Sign	Vgttn	Trrn	Sky	Person	Rider	Car	Truck	Bus	Train	Mcycl	Bcycl	mIoU
CLAN [2] - baseline	87.0	27.1	79.6	27.3	23.3	28.3	35.5	24.2	83.6	27.4	74.2	58.6	28.0	76.2	33.1	36.7	6.7	31.9	31.4	43.2
CLAN + Norm * ensemble	+/ 0.73	-/ 1.7	+/ 0.72	+/ 1.35	+/ 0.1	+/ 1.1	+/ 2	+/ 2.4	+/ 0.33	**+/ 8**	+/ 0.18	+/ 1.8	**+/ 0.9**	+/ 7.3	**+/ 2.32**	+/ 2.7	-/ 5.6	**+/ 0.93**	+/ 4.87	+/ 1.56
CLAN + Norm	**+/ 1.83**	**+/ 6.1**	**+/ 1.9**	**+/ 5.36**	+/ 0.94	**+/ 4**	**+/ 2.15**	**+/ 3.33**	+/ 0.34	**+/ 6.48**	**+/ 1.2**	**+/ 2.4**	+/ 0.1	**+/ 8.53**	+/ 0.26	**+/ 6**	-/ 5.7	-/ 0.8	**+/ 5.26**	**+/ 2.47**

Table 2. IoU table, The focus here is on the improvement made in each class comparing to the baseline architecture (CLAN [2]). With both ensemble and re-train methods. We can see that most of the classes had significant improvements.

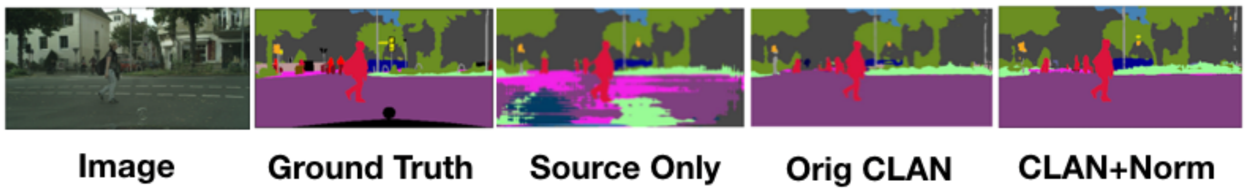

Figure 5. Prediction Visualisation, this picture compares between baseline model prediction and our improved model. We can see that CLAN+Norm prediction is more accurate and detailed according to ground truth label (second from left). Pay attention to the car in the background and the traffic light.

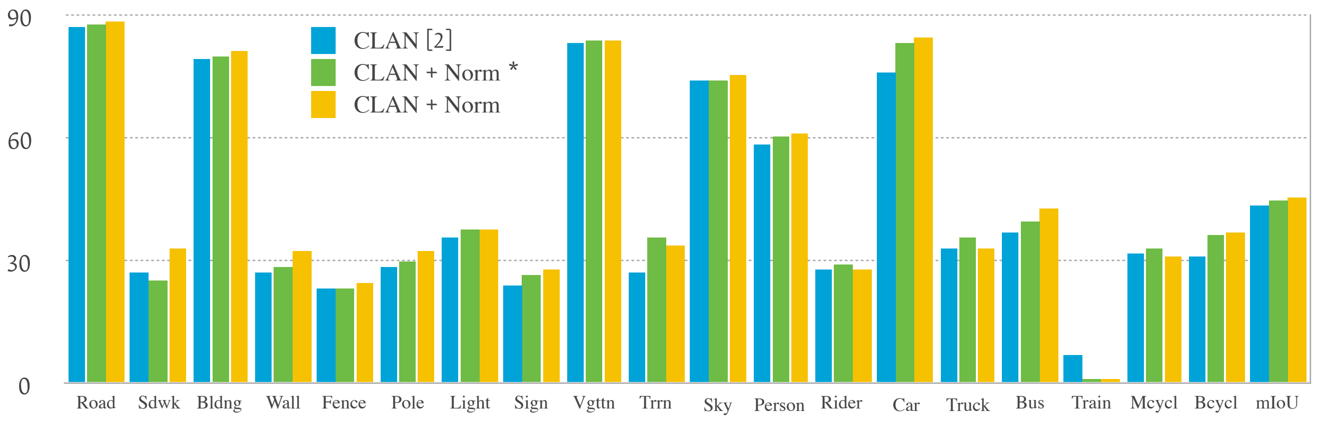

Figure 6. IoU plot, this plot compares results of each class between baseline architecture (CLAN [2]) and both ensemble and re-train improvement methods. There is major improvement in various classes, the main degradation is in 'Train' class which is relatively very low (may imply of class imbalance).

4. Future Work

Firstly, it would be interesting to test our methods on **more benchmarks** such as SYNTHIA to Cityscapes or Cityscapes to CrossCity. Secondly the Norm loss serves us well, but it would be interesting to find more **alternative losses for delicate feature alignment** or even to alter the norm loss in different ways. One interesting way could be trying to alter it in order to help fixing class-imbalance problems which definitely occur (for example take a look on **train** IoU results). Another interesting lead, is that **backbone upgrade is possible now**. While newest SOTA semantic segmentation architectures are heavy and computationally expansive, combining it with adversarial network components is almost impossible to train on standard GPU memory. Having said that, it would be interesting to test latest semantic segmentation models such as HRNet-V2 [6] with additional light computationally adapted step wise L_2 norm loss. We predict that the results would set a new benchmark in UDA for semantic segmentation, while training cost would still be reachable. Last but not least, direct feature space alignment using norm loss **might be also useful outside of UDA tasks**. Say for **generalisability of other problems**, such as supervised segmentation\ classification.

5. Conclusions

In this project, we have chosen to focus on a popular and competitive benchmark - UDA for semantic segmentation of GTA5 to Cityscapes. Although there was a lot of previous work done in the field, we've succeeded to present an innovative approach which provides three main contributions:

1. New method for boosting performances of previous and future UDA semantic segmentation architectures, which is done by targeting direct Feature Space alignment using L_2 - norm feature extraction enlargement.
2. A new UDA architecture for semantic segmentation, based only on direct feature space alignment using the norm loss. An architecture which is relatively computationally light, has adequate results (better than adversarial setups) and provides new knowledge on target domain which can be easily transferred to other methods via simple ensemble - this was demonstrated on both adversarial and non-adversarial baselines.
3. **SOTA** results on task of UDA semantic segmentation of GTA5 -> Cityscapes.

References

[1] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. arXiv preprint arXiv:1802.10349, 2018.

[2] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[3] Minghao Chen, Hongyang Xue, and Deng Cai. Domain adaptation for semantic segmentation with maximum squares loss. In Proceedings of the IEEE International Conference on Computer Vision, pages 2090–2099, 2019.

[4] Xu, R.; Li, G.; Yang, J.; and Lin, L. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In The IEEE International Conference on Computer Vision (ICCV), 2019.

[5] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced Wasserstein Discrepancy for Unsupervised Domain Adaptation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[6] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., Xiao, B.: Deep high-resolution representation learning for visual recognition. arXiv:1908.07919, 2019.

[7] Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A.A., Darrell, T.: CyCADA: CycleConsistent Adversarial Domain Adaptation. ArXiv eprints, 2017.