Few-shot learning on the fashion products dataset

August 21, 2020

1 Implementation details

1.1 The Fashion dataset

The dataset consists of high resolution product images. In addition, product descriptions and attributes are also included. Examples of of the images are shown below:

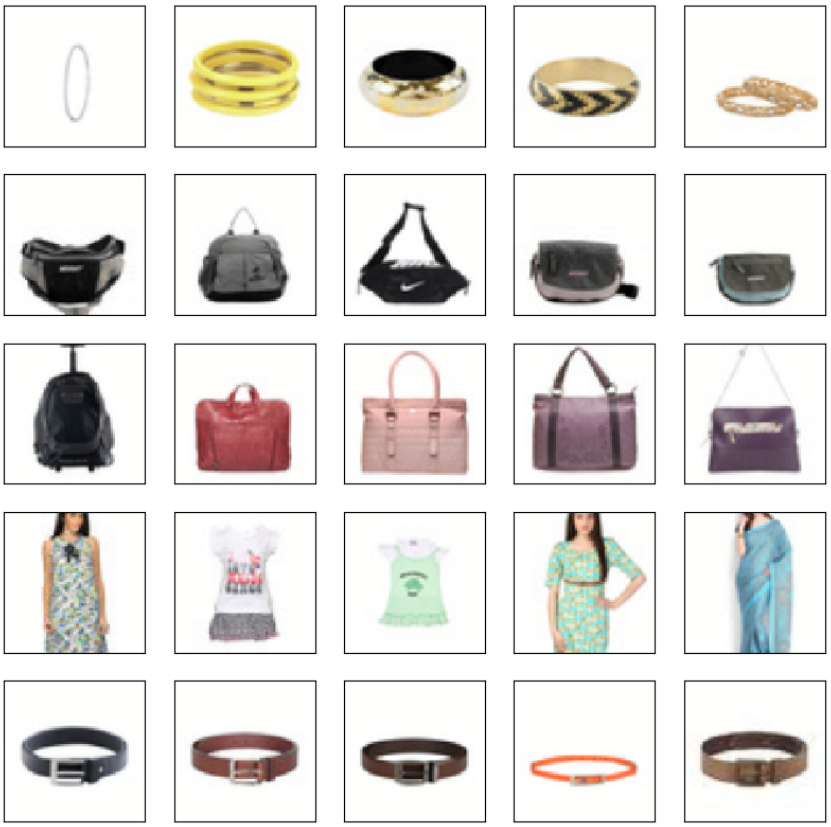


Figure 1: Multiple examples from multiple classes in the fashion dataset

The fashion dataset was split into two separate directories: background and evaluation. Both splits consist of disjoint classes to accommodate the few-shot learning paradigm. The background classes and the evaluation classes used in the experiments are listed in appendix A.1. There are 60 classes in the background split and 47 classes in the evaluation split. The dataset is severely imbalanced where number of examples range from 7000 for one class to 10 for another. The distribution of classes and their respective counts are illustrated in appendix A.2.

During the data preprocessing phase, some of the examples were grey scale (single channel) so their channels had to be tripled to fit the input of the network. In a few shot learning paradigm, data balancing is not crucial since the encoder sees approximately 1-5 examples of each class during training. However,

in the conventional learning manner where all data will be used to create mini-batches, data balancing or loss balancing is crucial.

Upon preprocessing and using the classes in A.1, the total number of examples for the training split is 25,044 compared to 19,243 for the evaluation split. The dataset has two variations: the small variation which has 80x80 pixel images and the large variation which has very high quality images which range from 150x200 pixels upto 3744x5616 pixels. Before training, when working with the small variation, data was not resized and was fed into the encoder with size (80,80) however the large images were resized to (160,160). Resizing to larger sizes was memory intense allowing lesser room for more experiments running concurrently on the training setup.

The data was subjected to various augmentation techniques to add representational robustness. Different techniques are discussed in later sections.

1.2 Augmentation Techniques

1.2.1 Constrained stochastic augmentation

In supervised image classification tasks, the number of images per class N_{orig} are very high. This exercise increases the robustness of visual representation since network parameters, which are discriminatively learnt, are optimized when exposed to lots of training examples. In few shot learning, the number of examples per class $N_{few} << N_{orig}$ are not enough to iteratively optimize the model parameters. Hence, attempts to alleviate this issue include increasing the number of training examples.

A straight forward approach is to apply constrained random augmentation to the limited number of example N_{few} hoping that random affine transformation alongside colour transformation would add more diversity to training data and hence improve the quality of the visual representations. This problem could formulated as follows:

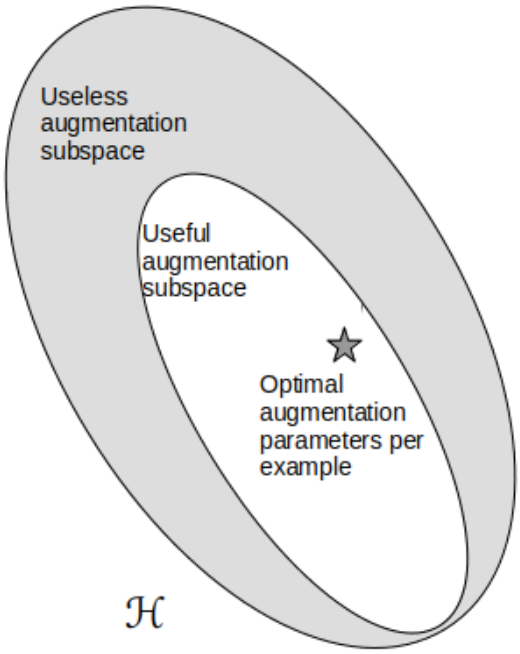


Figure 2: Augmentation subspace

The space of all possible image transformations $\mathcal{H}$ is constrained to subspace $\mathfrak{H}$ which all meaningful image transformations. In other words, the subspace $\mathfrak{H}$ includes all transformation that have been subject to a user's prior. For example, aggressive down scaling or upscaling of an image that leads to image disappearance or zooming to a single pixel respectively fall in the useless transformation space. On the other hand, moderate scaling which is limited by user prior by explicitly specifying an operational range

falls in the meaningful image transformation subspace. Straight forward augmentation is not conditioned on input data and as a result, parameters are stochastically sampled from $\mathfrak{H}$ regardless of the input.

Optimal transformation parameters for this augmentation scenario are undefined since there is no objective to help assess the quality of the transformation parameters. This augmentation method is widely adopted in both supervised and unsupervised visual representation learning. Meanwhile, this augmentation method is not crucial for supervised visual representation learning, it is fundamental to unsupervised learning methods. *Stochastic augmentation* is an important building block to unsupervised techniques [1][2][3] since it helps create correlated views of the same example which is impossible otherwise due to the absence of labels.

We employ this method in an online manner. In other words, a new set of transformation parameters are sampled from $\mathfrak{H}$ for each example in a batch. Eventually, the network would have been exposed to a multitude of data samples that have been transformed stochastically. In our augmentation experiments, we modify affine transformation parameters $[a, b, c, d, e, f]$ which introduce scaling, translation, rotation, shearing transformations. We also introduce color transformation methods that modify brightness, contrast, saturation and hue. Our configuration for both small and large variation is shown below in table 1:

Table 1: Different augmentation parameters employed for different dataset sizes

Size	rotation	translation	scaling	brightness	contrast	saturation
small	$-15°, 15°$	$0.1, 0.1$	$0.9, 1.4$	$0.9, 1.1$	$0.8, 1.2$	$0.8, 1.2$
large	$-15°, 15°$	$0.2, 0.2$	$0.7, 1.3$	$0.9, 1.1$	$0.8, 1.2$	$0.8, 1.2$

This augmentation table defines the subspace $\mathfrak{H}$ where the parameters are manually picked. Doing so, the encoder model will be exposed to the possibility of images being transformed by the above parameters and hence learns a more robust visual representation. Some stochastically augmented images are shown below:

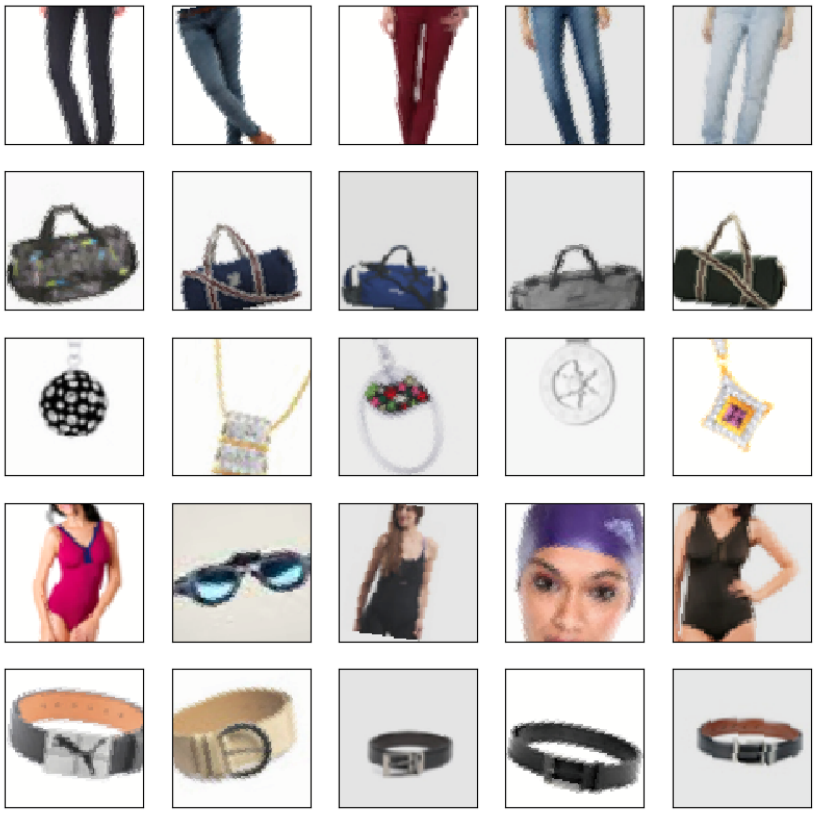


Figure 3: Stochastic augmentation of input samples

What if the augmentation parameters are learned to optimize a certain objective?

1.2.2 Unconstrained Spatial Transformer Network

A model could be employed to regress affine transformation parameters subject to a certain objective. These models are known as *Spatial Transformers*.[4] In brevity, the model optimizes the following function:

$$\hat{\theta} = \underset{\theta}{max} \sum_{i=1}^{N} L(y_i, \hat{y_i}) \tag{1}$$

where $\hat{\theta}$ is the optimal set of affine parameters that minimizes a certain classification loss L. In most multiclass classification problems, L is set to be a cross-entropy loss. y_i is the ground truth and $\hat{y_i}$ is the models prediction which we could be defined as:

$$\hat{y_i} = max \ \frac{exp(f(g_\theta(x_i)))}{\sum_j exp(f(g_\theta(x_j)))} \tag{2}$$

where $f(.)$ represents the parametric encoding function that maps the image from the visual domain to the embedding/representation domain such as $f : \mathbb{R}^{W \times H \times C} \rightarrow \mathbb{R}^D$. $g_\theta(.)$ is a parametric function that regress affine transformation parameters based on an input example x_i. For notational simplicity, we can assume $g(x_i)$ is the affine transformed example with affine parameters $\theta = [a, b, c, d, e, f]$. For most of the few-shot learning paradigms, the above equation employs a distance based loss as L which compares representations instead of classes. The distance is then passed to softmax function as in equation 2.

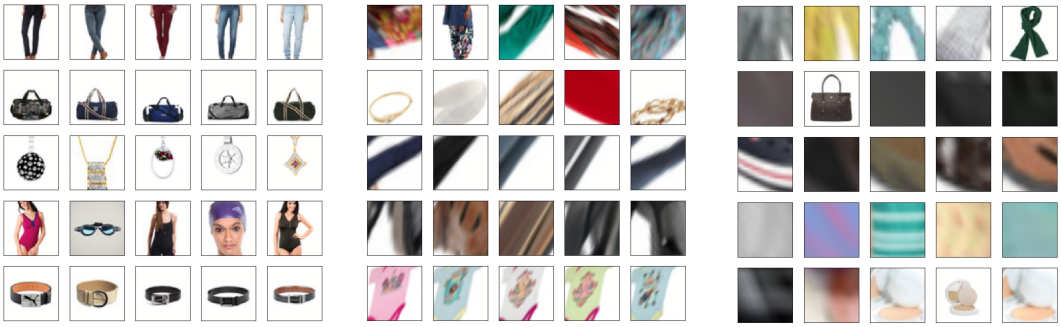
There are no parameter constraints applied to the above equation. In other words, the affine transformation function $g(.)$ operates within the full transformation space $\mathcal{H}$. This might lead to uncontrolled transformations. In a demonstration from the original spatial transformers paper, this was noticed in the colocalization experiment as shown in the figure below:

(a) Before (b) After

Figure 4: Unconstrained spatial transformer using meaningless affine transformation parameters

The spatial transformer in the original paper is trained in a weak supervision manner where the encoder function $f(.)$ is trained using direct supervision. Even though the original authors used a lower learning rate to update the spatial transformer parameters as a form of implicit regularization, this is a hyperparameter that implicitly constrains the transformation parameter space. In other words, when the transformer performs a meaningless transformation, the error produced by function $L(.)$ increases. This leads to parameter updates that punish such transformer behavior thus keeping the network within $\mathfrak{H}$.

When employing unconstrained transformation on the fashion dataset, similar behavior was noticed as shown below:



(a) Before (b) During (c) After

Figure 5: Unconstrained spatial transformer using meaningless affine transformation parameters on fashion dataset

In order to mitigate this issue, we will discuss different methods which aim to constraint $\mathcal{H}$ to $\mathfrak{H}$.

1.2.3 Manually constrained Spatial Transformer Network

What if all the parameters of the spatial network are clipped to mimic the constraint imposed in Table 1? In order to do so, we write the six affine transformation parameters in terms of fundamental operators such as rotation, scaling and translation. The affine transformation matrices of fundamental operators are defined in equations below:

$$t(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$s(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$r(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \sin(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

In terms of 3 - 5, the composite transformation can be expressed as:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \sin(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

which equates:

$$T = \begin{bmatrix} s_x \cos(\theta) & s_y \sin(\theta) & t_x \\ -s_x \sin(\theta) & s_y \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

And finally:

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Where $a = s_x \cos(\theta)$, $b = s_y \sin(\theta)$, $c = t_x$, $d = -s_x \sin(\theta)$, $e = s_y \cos(\theta)$ and $f = t_y$.
Therefore we can constrain parameters a, b, c, d, e, f as follows. Assuming that the angle θ is constrained in range:

$$\frac{-\pi}{2} < \theta < \frac{\pi}{2} \tag{9}$$

then:

$$s_{xmin} \cos\theta_{min} < a < s_{xmax} \tag{10}$$

$$s_{y_min} \sin(\theta_{min}) < b < s_{ymax} \sin(\theta_{max}) \tag{11}$$

$$t_{xmin} < c < t_{xmax} \tag{12}$$

$$-s_{xmax} \sin(\theta_{max}) < d < -s_{xmin} \sin(\theta_{min}) \tag{13}$$

$$s_{ymin} \cos\theta_{min} < e < s_{ymax} \tag{14}$$

$$t_{ymin} < f < t_{ymax} \tag{15}$$

Equations 10 to 15 define a hard constrained subspace $\mathfrak{H}$. All spatial transformer parameters are clipped to fall within these ranges. A basic clipping function that was used is shown below:

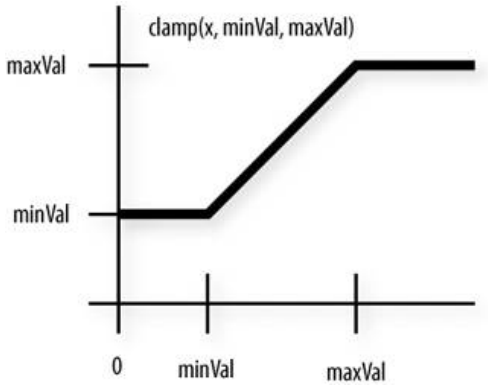


Figure 6: Hard clipping function

The derivative of the above image is non-zero only within the linear region. This property of the above constraining function is problematic for a learning algorithm that employs derivatives such as a neural network. In other words, the learning algorithm can get stuck at these edges since zero gradients translates to zero parameter update. An illustration of this can be shown below:

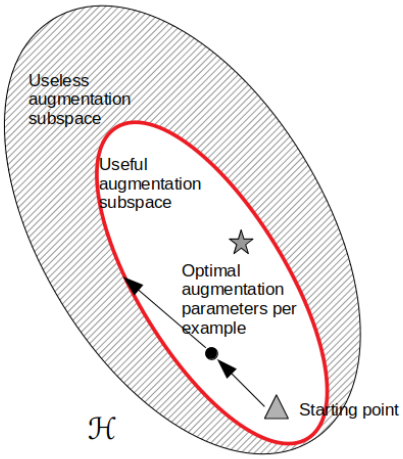


Figure 7: learning stuck at constraint values

The red outline in the above diagram exemplifies the constraints that were defined in equations 10 - 15. Due to the usage of the clipping function in figure 6 and due to the zero gradient problem, learning could get stuck at zero gradient regions (hatched). To prove the validity of this idea, we trained a manually constrained on a spatial transformer, the results subjectively demonstrate this behavior:

(a) Before (b) After

Figure 8: Constrained spatial transformer transforming all samples in batch with same edge parameters

The previous behavior results in the model using a deterministic set of edge affine parameters to transform all data. This results in lesser image variation which is sub-optimal making it a worse alternative in comparison to stochastic augmentation.

1.2.4 Regularization constrained Spatial Transformer Network

Following Model Agnostic Adverserial Augmentation for few shot learning [5], we create regularizer that pulls the model back into meaningful transformation area. The regulizer term can be described as:

$$L_{reg}(x_i) = \left\| \begin{bmatrix} a(x_i) & b(x_i) & c(x_i) \\ d(x_i) & e(x_i) & f(x_i) \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \right\|^2 \tag{16}$$

This can be illustrated as:

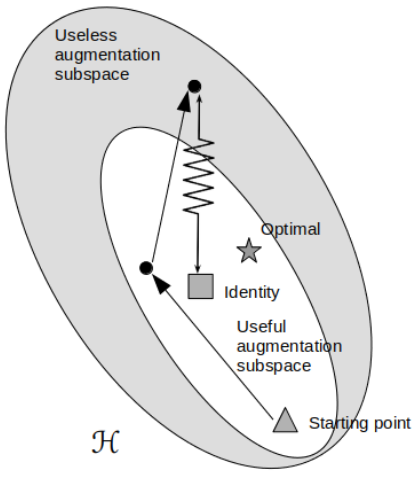


Figure 9: Spring effect

As demonstrated by the above figure, the regularization loss acts as a spring that pulls model parameters back to identity whenever they go far into $\mathcal{H}$. Meanwhile, this regularization is promising and has demonstratable results on three different few shot learning models, it is very hard to calibrate. The final form of the loss function when coupled with 16 is:

$$\hat{\theta} = \max_{\theta} \sum_{i=1}^{N} L(y_i, \hat{y_i}) + \lambda L_{reg} \tag{17}$$

Calibration includes finding the optimal λ value which represents the pulling power. If the value is high, we end up with trivial permutations of the identity transformation which don't add any variation in comparison to the non-augmented mode. On the other hand if the λ is low, the pulling power is low and hence the model parameters could dwell for within the unconstrained region thus contaminating training data.

2 Experiments and results

2.1 Setup

The specification of the machine are listed below:

- CPU: 12 x Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
- GPU: 2 x Nvidia K80

All models were trained for 200 epochs to ensure convergence.

2.2 Experiments

2.2.1 Prototypical networks results on fashion

Prototypical networks [6] were evaluated on both variations of the fashion dataset. The results of this evaluation are shown below:

Table 2: Prototypical networks evaluation on the small variation of Fashion products dataset k-train = 20 n-train = 1

k way	2	2	5	5	15	15
n shot	1	5	1	5	1	5
small	95.0	99.0	78.8	91.6	58.6	76.9
small (augmented)	95.0	98.0	79.0	91.8	55.6	77.6

The above results demonstrate a slight advantage of stochastic augmentation over non augmented results on some of the evaluation protocols. The slight increase could be attributed to augmenting both the query of and support of the training batch. We believe that augmenting only the training subset would be more beneficial and faithful to the actual evaluation protocol. During the above training, the number of classes (k way) was set to 20 and the number of shots (n shot) was set to 1. A high number was chosen since neural networks train discriminatively and hence if the number of distractors increase, training should be better. However, experiments were conducted on two protocols where the number of class (k way) was increased to 40 however the augmented version was not tested . The results are shown in table 3 below:

Table 3: Prototypical networks evaluation (k-train 20 vs k-train 40)

k way	2	2
n shot	1	5
k-train 20	95.0	78.8
k-train 40	93.0	77.6

Experiments suggest otherwise for very high number of classes (=40). We could speculate the problem became too hard for the network to progress and hence generalize better than the k-train = 20 scenario.

The results on the large variation for prototypical network training are shown below:

We believe that the lower results produced by augmentation could be attributed to same reason that we augment the query images. This makes the actual test a bit far from the training. We believe that upon fixing this, we should get an improvement. Another reason that on bigger images colour augmentation

Table 4: Prototypical networks evaluation on the large variation of Fashion products dataset k-train = 20 n-train = 1

k way	2	2	5	5	15	15
n shot	1	5	1	5	1	5
large	93.5	98.5	78.8	91.2	58.8	76.0
large (augmented)	92.5	97.1	76.2	88.2	53.8	71.9

distortion could be more noticeable since more the network kernels do more strides over the bigger image hence the effect could be compounded. These speculations should be examined to reach a more concrete conclusion. Future experiments include changing kernel sizes and not augmenting the query set during training.

2.2.2 Matching networks results on fashion dataset

Matching networks[7] were evaluated on the small variation of the fashion dataset while setting the Fully Conditional Embeddings (FCE) to false and using an L2 distance.

Table 5: Matching networks evaluation on the small variation of Fashion products dataset k-train = 20 n-train = 1

k way	2	2	5	5	15	15
n shot	1	5	1	5	1	5
small	95.0	99.0	79.4	91.0	59.1	74.9

It is apparent that the Matching network results are comparable with Prototypical networks on the small variation of the dataset. According to the original paper, using Fully conditional Embeddings should lead to considerable boost.

3 Future work

This section is divided into two subsections. In the first subsection, we list all the future experiments that would increase the robustness of our reported results. In the second section, we propose two novel techniques to constrain the spatial transformer search space.

3.1 Extra experiments

The following experiments should be conducted to increase the validity of the results:

- Augment support set only and evaluate prototypical networks and matching networks on fashion small and large.
- Evaluate Prototypical networks with unconstrained spatial transformers and doing the necessary calibration on fashion small and large.
- Evaluate Prototypical networks with a manually constrained spatial transformer on fashion small and large.
- Evaluate Prototypical networks with a manually regularized constrained spatial transformer network on fashion small and large.
- Evaluate Matching networks with fully conditional embeddings support on fashion small and large.

- Evaluate Matching networks with all the three variations of the spatial transformer on fashion small and large.
- Evaluate first order and second order MAML[8] on both fashion variations and all of the above.

3.2 Extra spatial transformer constraining techniques

3.2.1 Edge regularizers

In similar vein to [], we could punish the model whenever it uses $\mathfrak{H}$ edge parameters. A preliminary draft of the proposed regularizer bank could look like as follows:

$$L_{reg}(x_i) = \left\| \begin{bmatrix} G(a(x_i) - a_{max}) & G(b(x_i) - b_{max}) & G(c(x_i) - c_{max}) \\ G(d(x_i) - d_{max}) & G(e(x_i) - e_{max}) & G(f(x_i) - f_{max}) \end{bmatrix} \right\|^2 + \left\| \begin{bmatrix} G(a(x_i) - a_{min}) & G(b(x_i) - b_{min}) & G(c(x_i) - c_{min}) \\ G(d(x_i) - d_{min}) & G(e(x_i) - e_{min}) & G(f(x_i) - f_{min}) \end{bmatrix} \right\|^2 \tag{18}$$

where $G(x)$:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{x/2\sigma^2} \tag{19}$$

The σ parameter in the Gaussian function is configurable and can control the regularization extent from the barrier. This exercise could be better illustrated in the below image:

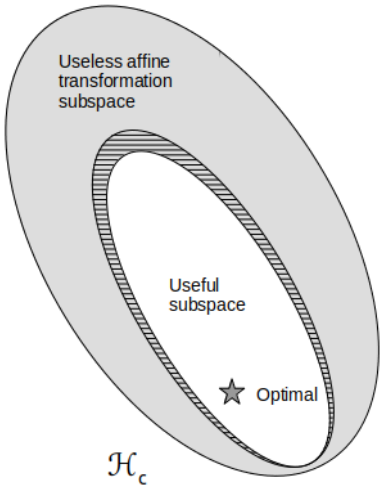


Figure 10: Gaussian edge regularization

3.2.2 Constraining via pretraining

In this method, we can pretrain the model without a spatial transformer until convergence. Upon convergence, a spatial transformer is embedded into the model and trained weakly while the model parameters are frozen. This would form a hard learnable constraint on the transformation space. When the spatial transformer is reaches a set of optimal parameters $\hat{\theta}$, we unfreeze all model layers and train the network in an end to end fashion. By doing so, we would have guaranteed excellent initialization of both models. An illustration of the thought process could be shown below:

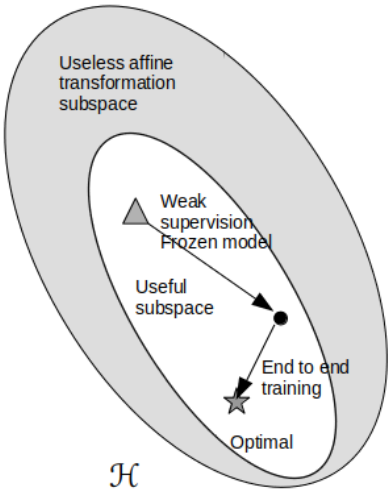


Figure 11: Pretrained constraining

4 Conclusion

We have investigated several augmentation techniques where the main learnable building block is a spatial transformer network. We have formulated the augmentation problem as a subspace optimization problem. Experiments to validate these ideas are pending. If these methods demonstrate any objective advantage, they could then be incorporated into a self supervised learning framework like SimCLR[1] or MOCO[2] for assisted augmentation. This method should result in maybe better results and faster training time. Some baseline results have been generated on Fashion small and large. We believe that this is a good starting point to embark on testing these ideas.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[2] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

[3] O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord, "Data-efficient image recognition with contrastive predictive coding," *arXiv preprint arXiv:1905.09272*, 2019.

[4] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, pp. 2017–2025, 2015.

[5] R. Jena, S. Sukanta Halder, and K. Sycara, "Ma3: Model agnostic adversarial augmentation for few shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 908–909, 2020.

[6] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, pp. 4077–4087, 2017.

[7] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, pp. 3630–3638, 2016.

[8] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

5 Appendix

A.1

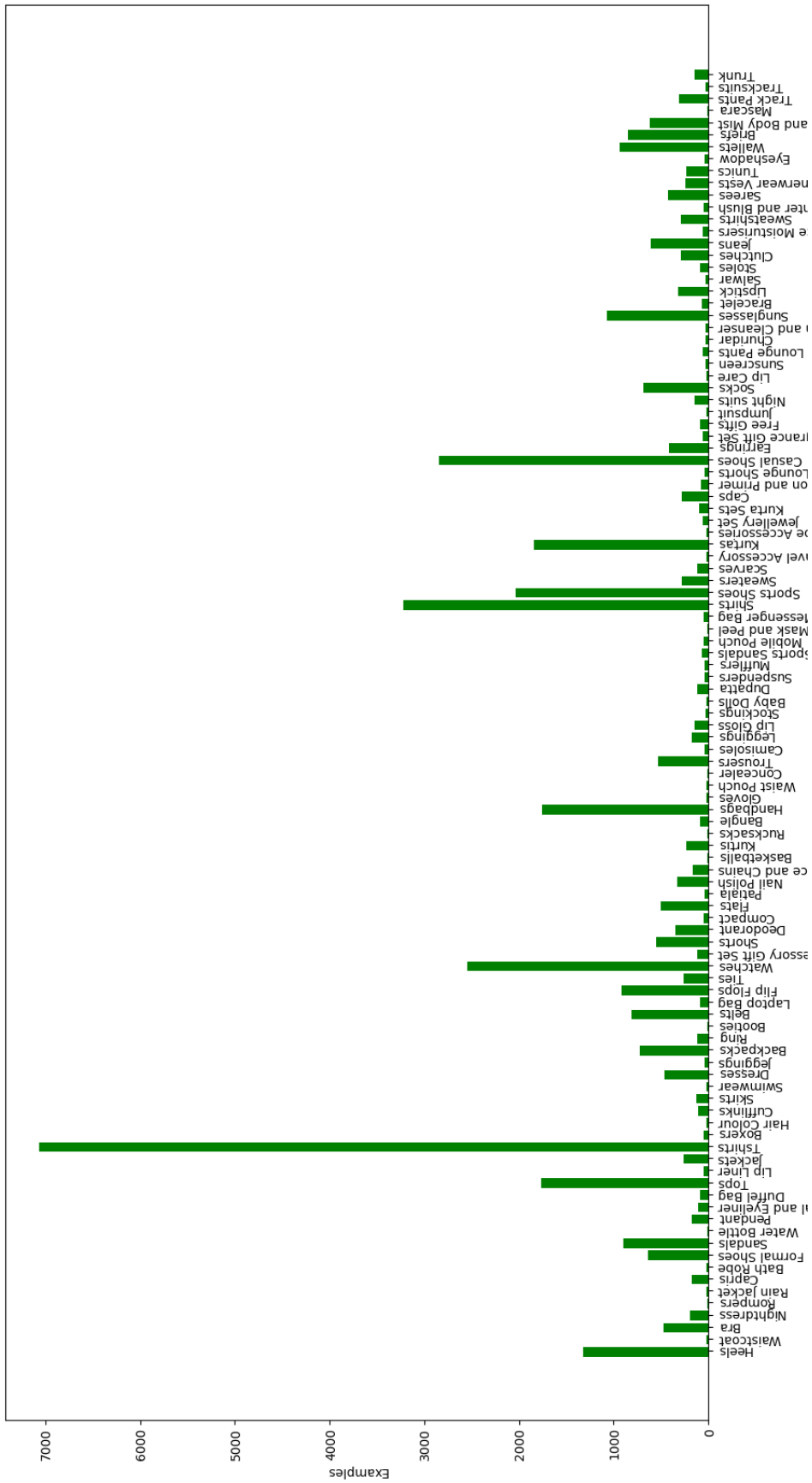
TRAIN SPLIT

Cufflinks
Rompers
Laptop Bag
Sports Sandals
Hair Colour
Suspenders
Trousers
Kajal and Eyeliner
Compact
Concealer
Jackets
Mufflers
Backpacks
Sandals
Shorts
Waistcoat
Watches
Pendant
Basketballs
Bath Robe
Boxers
Deodorant
Rain Jacket
Necklace and Chains
Ring
Formal Shoes
Nail Polish
Baby Dolls
Lip Liner
Bangle
Tshirts
Flats
Stockings
Skirts
Mobile Pouch
Capris
Dupatta
Lip Gloss
Patiala
Handbags
Leggings
Ties
Flip Flops
Rucksacks
Jeggings
Nightdress
Waist Pouch
Tops
Dresses
Water Bottle
Camisoles
Heels
Gloves
Duffel Bag
Swimwear
Booties
Kurtis
Belts
Accessory Gift Set
Bra

EVALUATION SPLIT

Jeans
Bracelet
Eyeshadow
Sweaters
Sarees
Earrings
Casual Shoes
Tracksuits
Clutches
Socks
Innerwear Vests
Night suits
Salwar
Stoles
Face Moisturisers
Perfume and Body Mist
Lounge Shorts
Scarves
Briefs
Jumpsuit
Wallets
Foundation and Primer
Sports Shoes
Highlighter and Blush
Sunscreen
Shoe Accessories
Track Pants
Fragrance Gift Set
Shirts
Sweatshirts
Mask and Peel
Jewellery Set
Face Wash and Cleanser
Messenger Bag
Free Gifts
Kurtas
Mascara
Lounge Pants
Caps
Lip Care
Trunk
Tunics
Kurta Sets
Sunglasses
Lipstick
Churidar
Travel Accessory

5.1 A.2


Examples
0
1000
2000
3000
4000
5000
6000
7000
Heels
Waistcoat
Bra
Nightdress
Rompers
Rain Jacket
Capris
Bath Robe
Formal Shoes
Sandals
Water Bottle
Pendant
Dufflel Bag
Tops
Lip Liner
Jackets
Tshirts
Boxers
Hair Colour
Cufflinks
Skirts
Swimwear
Dresses
Jeggings
Backpacks
Ring
Booties
Belts
Laptop Bag
Flip Flops
Ties
Watches
Shorts
Deodorant
Compact
Flats
Patiala
Nail Polish
Basketballs
Kurtis
Rucksacks
Bangle
Handbags
Gloves
Waist Pouch
Concealer
Trousers
Camisoles
Leggings
Lip Gloss
Stockings
Baby Dolls
Dupatta
Suspenders
Mufflers
Mobile Pouch
Mask and Peel
Shirts
Sports Shoes
Sweaters
Scarves
Kurtas
Kurta Sets
Caps
Casual Shoes
Earrings
Jumpsuit
Free Gifts
Night suits
Socks
Lip Care
Sunscreen
Lounge Pants
Churidar
Sunglasses
Bracelet
Lipstick
Salwar
Stoles
Clutches
Jeans
Sarees
Tunics
Eyeshadow
Wallets
Briefs
Track Pants
Tracksuits
Trunk