Deep Semantic Mapping Between Images

Adan Häfliger

***Abstract*—In this project we explore a technique to extract information from layers of a deep convolutional neural network trained on image recognition. We run two or more corresponding images in the same neural network to get matching representation of each image by extracting, for each image, one activated layer. We show that we can construct semantic soft masks matching the same element in two images using the activated layers. We can also produce image segmentation based on known similar images if their segmentation is known. The technique mainly consists of using correlations between the activated layers to extract matching semantic parts. We show two use cases where our method is applicable: Style Transfer and Multiple Image Labelling. In Style Transfer we can condition the style loss with several mask pairs, each pair matching the same element in both image. This makes the style transfer more localized. In Multiple Image Labelling we use our method to label each pixel by using the strongest correlation between layers of every (image to label, image) pair. We then validate our result on the Pascal VOC 2012 val dataset.**

I. Introduction

Deep Convolutional Neural Network trained for image recognition like VGG19 [12] are able to classify images as belonging to one of a thousand classes. To perform this classification, we have the intuition that the network must produce a representation of image content that is semantic. It is known that early layer activates for simple shapes and the later ones activate on more abstract elements. The main assumption of our method is that when we run an image on these networks, each channel detects the same parts for most images. Using that property and deep enough layers, we can compare how the same layer, run on two different image correlates to produce masks. The method consists of computing a correlation matrix between each layer that we call the Affinity Matrix and to apply Non Negative Matrix Factorization to it. That way we get a subset of the most meaningful component that can reproduce the Affinity Matrix. Those component can be seen as masks matching the same semantic elements between two images.
The benefits of our method is that we can extract these masks directly from a pre-trained model that was not meant to perform segmentation or semantic mapping. Also it is easier to produce datasets for image recognition compared to segmentation. Our method also work for classes that are not recognizable from the trained network.

II. Related Work

With our method we can construct co-segmentation between two images. Co-segmentation consists in dividing several images into regions corresponding to the same classes. Given the different scale and brightness of images, developing an algorithm that adapts to most cases can be difficult. As such, there are supervised approaches where user input is necessary such as [1] and unsupervised approaches that performs co-segmentation automatically like in [2]. In our case we use a neural network that is already good at dealing with different images to perform our co-segmentation.

One of our application, style transfer, is improved by using co-segmentation. Our work relates to [6] which introduced style transfer using convolutional neural network feature maps but we will also reference [9] which is where we took several ideas.

We can also perform segmentation directly. To validate our method for segmentation we compare our accuracy with a baseline from [10] on the VOC 2012 Challenge see [5]. Our method is different and might not completely abide by the rules of the competition since we extract our segmentation from a network trained on ImageNet [4] which is outside of the VOC Challenge, nonetheless this comparison gives a good idea of the performance of our method.

III. Method

A. Affinity Matrix

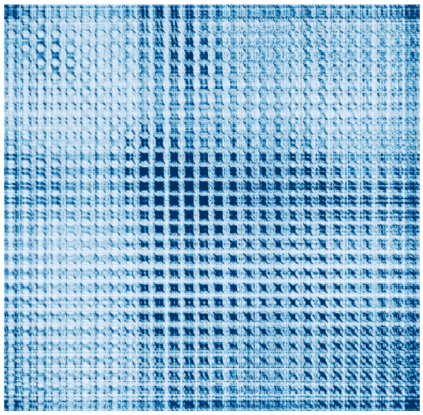

Fig. 1. Visualization of the affinity matrix computed from two layer of a deep convolutional neural network - contrast is skewed to better show structure

Here we will describe the main component of our technique and how to compute it from VGG layers. Assume we have the following:

- $L_1 \in (h_1 \times w_1 \times C)$ feature map activation of 1st image
- $L_2 \in (h_2 \times w_2 \times C)$ feature map activation of 2nd image

We flatten L_1 and L_2:

- $L_1 \in (h_1 \cdot w_1 \times C)$
- $L_2 \in (h_2 \cdot w_2 \times C)$

Then we compute the matrix:

$$A = L_1 \times L_2^T \tag{1}$$

Therefore $A \in (h_1 \cdot w_1 \times h_2 \cdot w_2)$. A visualization of this matrix can be seen in figure 1. This matrix quantifies how each pixel is correlated with every other pixel of both activation. For example the value at position (1,1) is the the value that represents how correlated the first pixel of the activation of the first image is with the first pixel of the activation of the second image. This matrix is useful because each layer activates on certain shapes/parts usually wherever these parts are in the image. in figure 2 you can see the input of the affinity matrix (what it is computed from), in that case the two images are close so each layer channel are remarkably similar. The Affinity Matrix is only made of positive values since we compute it from a ReLU activation function in the case of VGG19.

The complexity of computing A grows with the square of each dimension of the images. That means that for deep layer of small spatial size it is almost instantaneous but it can take minutes for the largest layers. That also means that the computation time gets longer as we increase the size of the input images but with 4 pooling operation when taking the deepest layer, we divide each dimension by 16 therefore the size is usually not a problem.

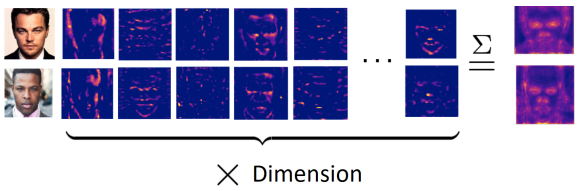

Fig. 2. A visualization of the input to the Affinity Matrix computation. The activation of each channel of two layer (dim=128) on two faces with the sum of the parts showed - this shows how the activation is symmetric (between the two images) in the activated values.

B. Style Transfer Introduction

In this application, the goal is to transfer the style of an image into a content image to produce a stylized version of it. This broad definition can imply several transformation on the content image. Global style transfer algorithms such as [11] effectively match the global statistics of both image to perform global color shifts. In our case we are interested in local style transfer based on the feature maps of a deep convolutional neural network as introduced by [6] where they technique is used for painterly style transfer .

The method consists of performing iterative optimization with an optimizer called L-BFGS-B [13] to constraint a generated image G to minimize some loss. All the loss terms are computed from the layer of G run into a convolutional neural network and different internal representation are used to compute these losses. There are 2 main loss term:

- The content loss L_c
- The style loss L_s

Let's assume a layer with N_l distinct filter has N_l feature maps each of size M_l. So we can store all the response of all the filter of layer l in $F^l \in \mathbb{R}^{N_l * M_l}$ where F^l_{ij} is the activation of the i^{th} filter position j in layer l. Also G^l and F^l are the feature maps of the response from a layer l of the generated image and the content image respectively.

L_c constraints G to have an identical feature map response (usually l = conv4_1 is used) to the content image. It's computed as follows:

$$L_c = \frac{1}{2}\sum_{i,j}(F^l_{ij} - G^l_{ij})^2 \tag{2}$$

L_s constraints G on several layers such that their gram representation with the style image are identical. the gram operation for matrix R is defined as: $R \cdot R^T$. Let's say that S^l is the feature map of the response from a layer of the style image. A style term for layer l is computed as:

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(Gram(S^l_{ij}) - Gram(G^l_{ij}))^2 \tag{3}$$

And the total style loss is then:

$$L_s = \sum_{l=0}^{l} w_l E_l \tag{4}$$

where w_l is a weight that controls the ratio between L_c and L_s. In our revised method explained in section IV-B1 this term is adapted.

A possible problem with the style loss is that the gram matrices are computed over the entire image. As explained in [9] It means that the gram representation is "limited in its ability to adapt to variations of semantic context". This is where our method and our co-segmentation technique is introduced.

From the Affinity Matrix we will see in the next section how we can compute mask pairs that can be used to condition L_s. In our case let's assume we have K mask pairs mapping element in both the content image and the style image. Also assume M_{fk} is the mask for content image and M_{sk} is the mask for the style image. We change each loss term as:

$$E_l = \sum_{i,j}\sum_{k=0}^{K}(Gram(S^l_{ij}M_{sk}) - Gram(G^l_{ij}M_{fk}))^2 \tag{5}$$

Which will map some parts of the gram. Now let's see how we compute our masks.

C. Non-Negative Matrix Factorization to extract masks

1) Motivation and Theory: We want to extract the most meaningful parts of the Affinity Matrix to produce mask pairs that match the same content. Two algorithm were considered: Principal Component Analysis (PCA) or Non-Negative Matrix Factorization (NMF). We use result from [8] to assert why NMF is better in our case. First consider PCA. It produces holistic representation with negative weights which are hard to use as masks. Also the NMF representation hopefully is part based which matches our use case especially because we have the property that A is only composed of positive values. In practice that makes NMF a good fit for our method.

NMF is cast as an optimization problem where we want to find two matrices L and R such that $LR^T \approx Q$ where we control how many parts we want with an hyperparameter K (which is the same symbol as the number of masks in section III-B since it will come from that). L and R are computed by iterative optimization with non-negativity constraints.

2) Procedure: By applying NMF to the Affinity Matrix with hyperparameter K, we obtain two matrices:

- $L \in (h_1 \cdot w_1 \times K)$
- $R \in (h_2 \cdot w_2 \times K)$

that approximate A. We can reshape these matrices to get K masks pairs of the dimension of the original activated layer. This means that the masks are always down-scaled by a factor of $2 \times i$ where i can be up to 8 (for each dim) for the deepest layer. This means that an original image of size (448, 448) will produce a mask of size (28, 28).

In general we found K in the range [5-15] to be good depending on how many parts we want, but we lack a way to validate what K to use. A large K gives more precise mapping but also amplifies noise and one object can be separated into several masks.

3) Refinement - Orphan mask: A shortcoming of NMF applied to A is that it produces imbalanced masks (we used sklearn's[1] NMF method) also the masks are overlapping and not all pixel are matched (sometimes more than half the pixel of the original image are not present in any of the masks). Given those characteristics we considered several normalization schemes and one improvement.
Normalization wise, We can normalize L or R directly but that makes certain mask almost zero because one mask pair can have higher values overall than another mask pair. The fact that one pair as higher values doesn't mean that the other mappings are meaningless. Another approach is to normalize each $L[k]$ $R[k]$ pairs which solves the above problem but it amplifies noise too much.

To solve the problem of pixels that are never matched, we introduce an orphan mask which maps pixel that are not matched in L and R. Its computation follows from the way we process the mask in the end.

Instead of the two above approaches we do the following: First we add a new mask pair which has all values as the mean of L, R respectively. Secondly we compute the softmax of each pixel in the dimension of the masks. So for L the first pixels of each mask of L will be softmaxed together. Since we included the mask with the mean, all pixel will be mapped, they won't overlap and the mean mask acts as an orphan mask. You can see one such mask in figure 3

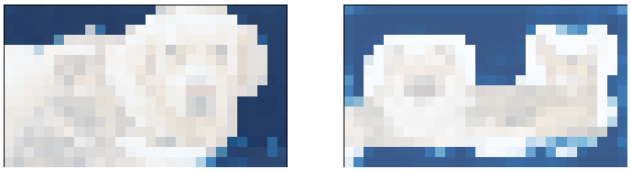

Fig. 3. Example of the orphan mask after the whole procedure - K = 10 in that case. The orphan mask segments the background quite well

But this comes with a price, indeed the mean mask values after softmax can be too small with respect to all the other values (remember L and R aren't balanced) but we can control the strength of the softmax with a "temperature" parameter β. This simply consists in dividing all the values by β before applying the softmax. The following equation show the operation for pixel i, j:

$$L'_{i,j} = Softmax([L[0]_{i,j}, L[1]_{i,j}, \dots L[K]_{i,j}]/\beta) \quad (6)$$

This let's us balance the the mask and spread out the values. Usually β is around 0.01. We go through this transformation to get better masks for style transfer.

In figure 4 you can see one of these masks and how it is semantic in the sense that two similar object are mapped in both images. Of course the technique can fail or the same element can be cut off in two different mask as in figure 4 where the tire of the car in Image L image was matched in another mask with other part of the tire of image R. Some masks also are harder to interpret by a human hinting that maybe some mask might be random correlation in both images instead of proper semantic objects.

Fig. 4. Two images with one of their corresponding semantic mask computed from the deepest layer (conv5_4) and K = 10 - in this example we can see that the mask pair matches the tires of both the plane and one of the car but it fails to get the tires of the car of Image L as it gets matched in another mask pair.

[1] http://scikit-learn.org/stable/

We also tried to run our masks through CRF [7] as a post-processing step but given how our masks are, the CRF doesn't work well most of the times. This effect is shown in figure 5. Note that we did not try to specifically fine tune the hyperparameter of the CRF.

Fig. 5. The effect of applying CRF to our masks. Given the usual small size of mapping some mask are simply removed. CRF was used as a black box with default parameter and can probably be fine-tuned to work better.

D. Model Consideration

We transformed a VGG19 network to be fully convolutional by removing the fully connected part at the end such that we can apply our method to any image size and dimension. We should also keep in mind that VGG19 was trained on Imagenet [4] where images are of size (224, 224) and there are one thousand classes to classify. This means that in some cases, we extract mask for categories that are not in Imagenet but it still works.

We also considered using a batch normalized version of VGG19 but we found that the produced masks were noisier as it affects the affinity matrix in the first place. The only investigation we did was to see that the layers themselves were noisier which could explain why our method doesn't work too well.

E. Image size and depth

There is also the question of what layer from VGG to use which leads to a size trade off. The deepest layer have better semantics but they have also lower dimension. You can see the effect of increasing depth in figure 6. From our experience taking the deepest layer is always recommended.

We also experimented with atrous convolutions [3] which is a method to perform a convolution while adding space in the kernel used for the convolution. The amount of space between each row and columns is defined by a rate parameters (normal convolution has rate = 1). This let's perform the same operation as if the prior layer was max pooled by removing that prior max pool and instead doubling the rate. That trick can double the dimension of our mask if we do it once since we remove a pooling operation but the downside is that it add holes in the layer which impacts the produced masks.

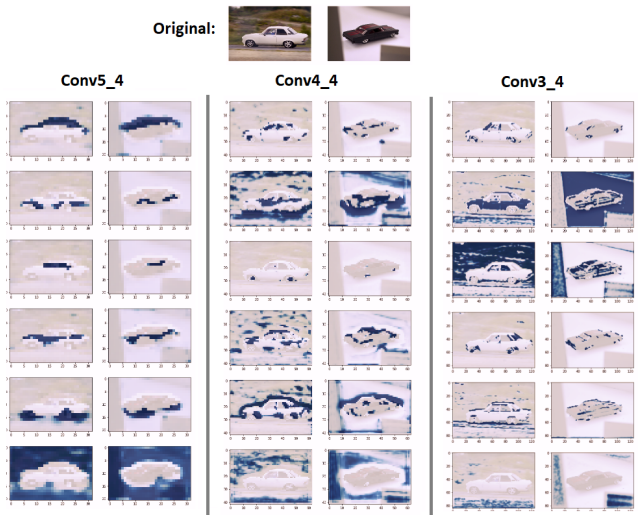

Fig. 6. Effect of decreasing layer depth when computing mask pairs - spatial size is reduced but semantic mapping is greatly improved

Overall We can consider applying the same technique on more recent network architecture or in a VGG like network when a bigger (dimension) ImageNet dataset is produced.

IV. Applications and Results

A. Multiple Image Labelling

1) Theory and method: Let's define the problem. We have an image S belonging to a class that is known and we have several other images C in the same class with segmentation masks given. We want to segment S using the other images and their masks. For this application we use the Affinity Matrix directly.

The idea is to use A to label each pixel of S by choosing the value that correlates the most to one of the images of C.

Let's describe the method with only one image in C and its segmentation mask for each classes C_m. The first step is to apply a softmax to A row wise. That picks which layer slice pair is the most important for that pixel. Then we compute the dot product of each row with the flattened mask of C.

The dimension works out like so:

- $A \in (l_1 \times l_2)$
- $C_m \in (h \cdot w \times N)$

where N is the number of labels. We downscale and flatten $C_m \in (l_1 \times N)$. We can then compute for each row of A the dot product of the row with C_m so for the first row:

$$A[i] \in (1, l_1) \tag{7}$$

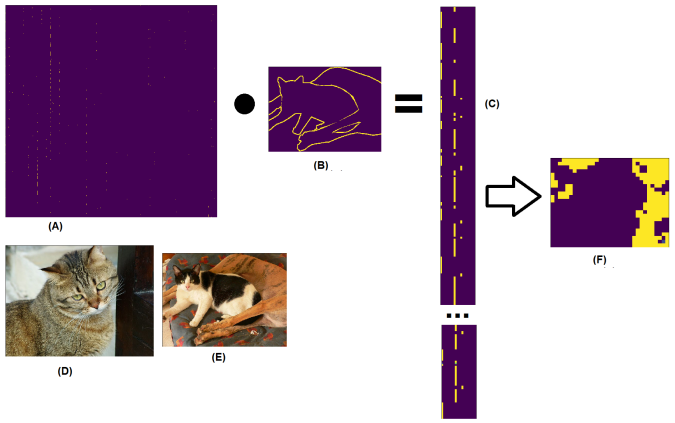

Fig. 7. (A) Affinity matrix with softmax applied row wise. (B) Mask used for labelling. (C) Segmentation computation using the affinity matrix. (D) Image to segment. (E) Image used to compute the affinity. (F) Final segmented image (before applying crf)

we get the label for the first pixel:

$$P_i = A[i] \cdot C_m, P_i \in (1 \times N) \tag{8}$$

If we do this for each row we get our segmentation amongst all the classes.

We can extend C_m by stacking the masks of more images and A by stacking the Affinity Matrices computed from each $S : C[x]$ pairs and the softmax will do its job to prioritize the image with the strongest correlation for a given pixel and chose its segmentation.

At this point the dimension of the predicted segmentation is the same dimension as the layer from A therefore we need to upscale it to the original dimension of the image. We use CRF [7] to re-size and better match the image. This gives better accuracy than up-scaling directly. The CRF values can certainly be fined tuned (again we use the default recommended parameters) and other method could be considered. In figure 7 you can see an image with an overview of the pipeline.

2) Results: An example of the application is shown in figure 8 where we compare our mask with and without applying CRF. We also include the effect of increasing the number of images at the same time. As expected the segmentation gets more and more accurate and the trade-off for adding more image is quite small since the time to compute the affinities at such size is negligible compared to applying the CRF.

We validated our results with the PASCAL VOC validation dataset. You can see on table I the per class average precision which is computed the following way: $tp/(tp + fn + fp)$ (ROC curve). You can also see the values from [10] (we took the values from the method that doesn't use segmentation priors) to get an idea on how our method performs (tagged as BL).

TABLE I. RESULT ON PASCAL VOC SEGMENTATION VAL DATASET

#img	plane	bicycle	bird	boat	bottle
1	0.34	0.07	0.26	0.28	0.15
2	0.4	0.06	0.32	0.29	0.19
4	0.42	0.05	0.35	0.29	0.19
8	0.45	0.07	0.36	0.35	0.22
16	0.46	0.08	0.38	0.36	0.22
BL	0.48	0.21	0.31	0.28	0.35
	bus	car	cat	chair	cow
1	0.45	0.2	0.46	0.07	0.43
2	0.5	0.22	0.53	0.08	0.46
4	0.49	0.27	0.55	0.09	0.49
8	0.53	0.31	0.55	0.1	0.5
16	0.54	0.29	0.58	0.1	0.5
BL	0.51	0.55	0.53	0.07	0.56
	table	dog	horse	bike	person
1	0.17	0.41	0.41	0.38	0.16
2	0.19	0.44	0.47	0.4	0.21
4	0.22	0.43	0.51	0.4	0.26
8	0.21	0.45	0.52	0.43	0.27
16	0.26	0.46	0.53	0.45	0.29
BL	0.2	0.54	0.50	0.40	0.39
	plant	sheep	sofa	train	tv
1	0.07	0.40	0.24	0.39	0.28
2	0.15	0.46	0.29	0.47	0.33
4	0.17	0.47	0.21	0.50	0.37
8	0.18	0.47	0.25	0.48	0.33
16	0.21	0.49	0.24	0.50	0.37
BL	0.28	0.52	0.25	0.33	0.46

Something to note is that there is randomness involved in the score we get, because we pick random images from the validation set that include the class from the image we try to segment. And these picked images change for each image that we segment. With more time we could improve that procedure to something more robust, or averaging several runs.

Also we could compare our method with techniques that use multiple image to perform their segmentation

B. Style Transfer

1) Revised method and discussion: In [9] they extend [6] by adding another loss term: The Photo-realistic loss L_m. L_m is constraining G to be represented by "locally affine color transformations of the input to prevent distortions" [9] using the Matting Laplacian of the content image. We suggest reading the mathematical definition of how this loss is formulated in [9] where it is well explained.

Our revised style transfer method is similar to [9], we

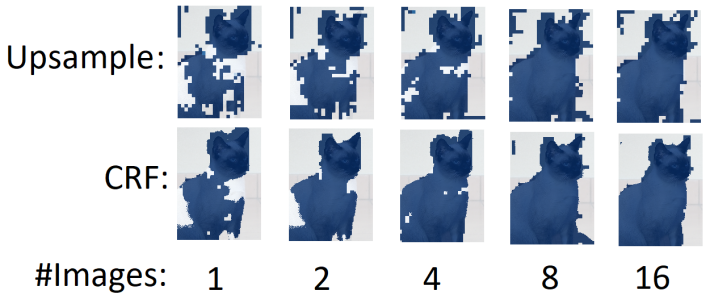

Fig. 8. Effect of using crf and changing the number of images in MIL - given the small size of the layer the affinity matrix computation is negligible compared to the CRF pass. Therefore we can in theory add more and more images.

include the photo realistic style term and we condition the style loss terms as explained in section III-B. The difference is that in our method, the network used to compute the different losses term is the same one that we produce the masks from. We will focus the discussion on the specifics of our pipeline.

The main interest of our method is that it doesn't require any external segmentation to localize the style transfer since it is directly computed from the same network used for the style transfer. Another advantage is that we can change the number of component of the segmentation by varying K. Indeed if we had fixed masks and labels, we could only get segments for these known label whereas our method can find K masks pairs. For example an background mask might cover various elements that should be mapped differently. Of course depending on K some masks pairs are not useful or too small. But this means that if there are elements for which no segmentation exists, our method could still be used although we don't control what segment are produced.

We condition L_s using our masks by multiplying the gram of each layer chosen for the style loss by its corresponding mask. Note that the loss is also multiplied by the mean value of the mask of the content image to account for varying mask sizes.

The other important point is to discuss the hyperparameters used when optimizing. Each loss term is multiplied by a constant which drives how important each factor is. There is no way of fixing these constants but experimenting. Also the best values change depending on the image structure. A typical set of constants for each loss term is:

- L_c : 100
- L_s : 0.02
- L_m : 2500

But what matters is the ratio between these constant with respect to the learning rate of the optimizer. That can be tricky to adjust and a major step forward would be to find good parameters automatically. And we usually want to run a 1000 iterations depending on the image size. Also for some images it's better to drop the orphan masks altogether to produce a better effect.

In [9] they mention that they use a two step optimization process where they first disable L_m in the first pass and then optimize again with all the loses. In our tests this procedure doesn't improve the result much.

2) Results: In order to get a good grasp of difference between methods we show in figure 9 the style transfer done in 4 different ways. One is without any conditioning on the style loss and no L_m. The second one is without masks but with L_m. The third one is there to experiment with VOC Pascal segmentation since it is available to us at this point, and the last one uses our masks. The constants are fixed for all method which can give advantages to some methods. It is hard to get a fair comparison since each method is dependant on the parameters fed. Figure 9 is meant to explain the methods and show their effect and not highlight the best use case of our technique. In the appendix we include more example as well as fail cases.

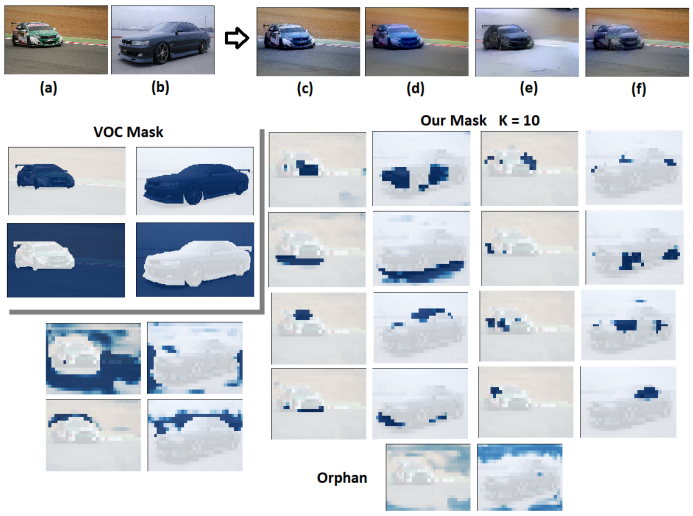

Fig. 9. Comparison of 4 style transfer method with the same constants. (a) content image. (b) style image. (c) basic method. (d) basic method including L_m. (e) using VOC Pascal mask. (f) using our mask.

We can also use a subset of all mask pairs selected by a human. On figure 10 you can see the effect of two mask pairs matching the cat's faces. We could create a semi-supervised method where a human picks the best mapping.

V. Future Work and Conclusion

In this work we showed how we can repurpose a pretrained model on image classification to extract dense semantic correspondences masks. There are in turn used to localize

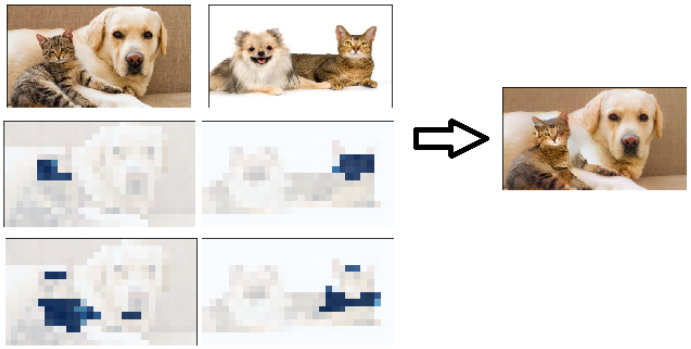

Fig. 10. When applying our style transfer using only a subset of our mask pairs - In that case we transfer the style of the cat's head, as can be seen from the cat's mustache for example

Style Transfer. We also show how to perform segmentation based on known similar images with their segmentation. This work can be expanded by testing the same method on different architectures potentially giving insight in their intrinsic properties.

VI. Implementation details

The project was coded with Tensorflow and Python and the code is available in Github.

References

[1] D. Batra et al. "iCoseg: Interactive co-segmentation with intelligent scribble guidance". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 2010, pp. 3169–3176. DOI: 10.1109/CVPR.2010.5540080.

[2] K. Y. Chang, T. L. Liu, and S. H. Lai. "From co-saliency to co-segmentation: An efficient and fully unsupervised energy minimization model". In: *CVPR 2011*. June 2011, pp. 2129–2136. DOI: 10.1109/CVPR.2011.5995415.

[3] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *CoRR* abs/1606.00915 (2016). arXiv: 1606.00915. URL: http://arxiv.org/abs/1606.00915.

[4] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[5] M. Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[7] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

[8] Daniel D. Lee and H. Sebastian Seung. "Learning the parts of objects by nonnegative matrix factorization". In: *Nature* 401 (1999), pp. 788–791.

[9] Fujun Luan et al. "Deep Photo Style Transfer". In: *CoRR* abs/1703.07511 (2017). arXiv: 1703.07511. URL: http://arxiv.org/abs/1703.07511.

[10] Pedro H. O. Pinheiro and Ronan Collobert. "Weakly Supervised Semantic Segmentation with Convolutional Networks". In: *CoRR* abs/1411.6228 (2014). arXiv: 1411.6228. URL: http://arxiv.org/abs/1411.6228.

[11] Erik Reinhard et al. "Color Transfer Between Images". In: *IEEE Comput. Graph. Appl.* 21.5 (Sept. 2001), pp. 34–41. ISSN: 0272-1716. DOI: 10.1109/38.946629. URL: http://dx.doi.org/10.1109/38.946629.

[12] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[13] Ciyou Zhu et al. *L-BFGS-B - Fortran Subroutines for Large-Scale Bound Constrained Optimization*. Tech. rep. ACM Trans. Math. Software, 1994.

APPENDIX A
IMAGES APPENDIX

Fig. 11. More MIL examples - including people and bicycle which is harder to segment by our method according to our validation score. Images chosen randomly.

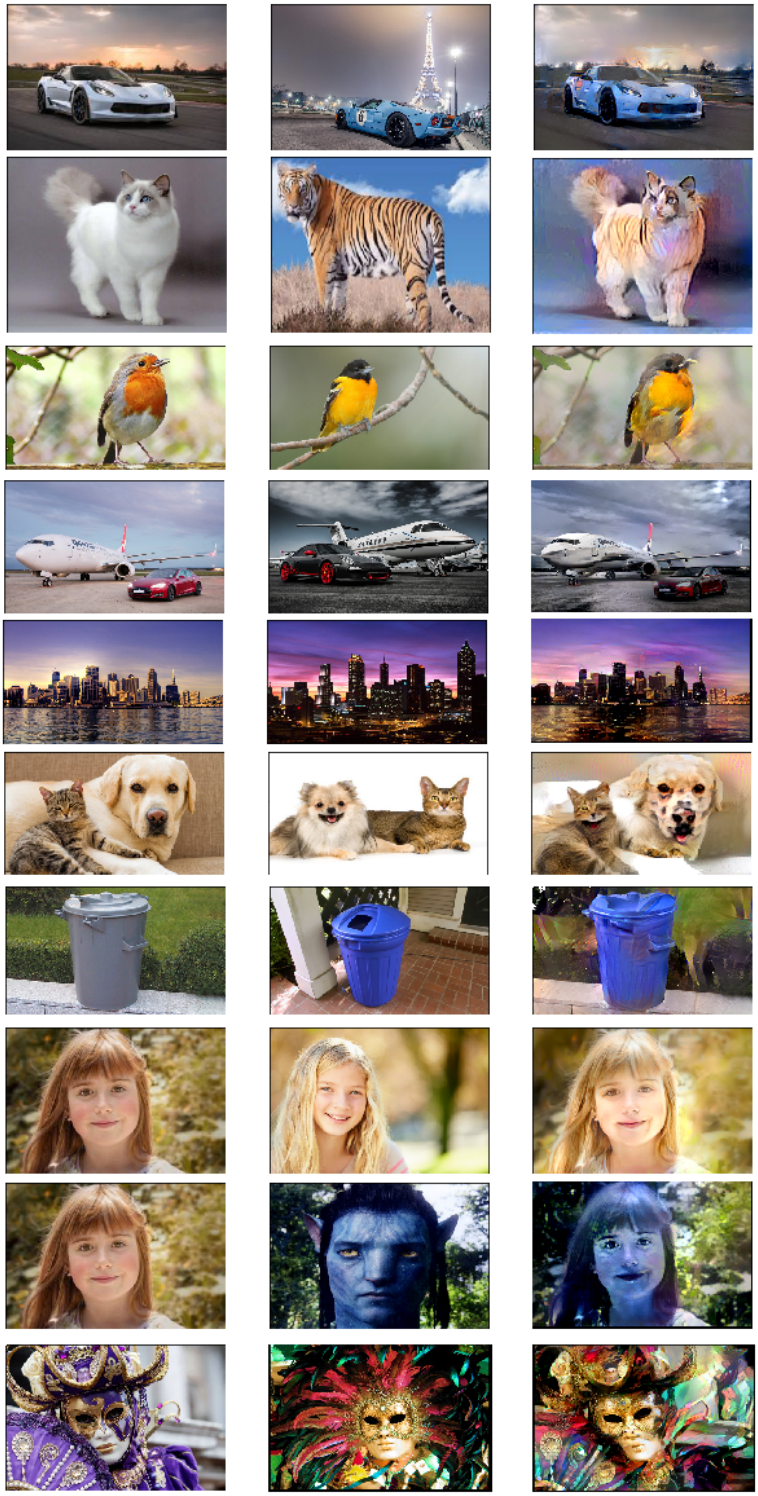

Fig. 12. More Style transfer examples, almost all parameters are fixed except for the dog and cat image and the carnival mask (more distortion gives interesting effects). Constants for each loss term are $L_c = 1000$, $L_s = 0.01$, $L_m = 10000$, $\beta = 0.01$ and K was either 4 or 7

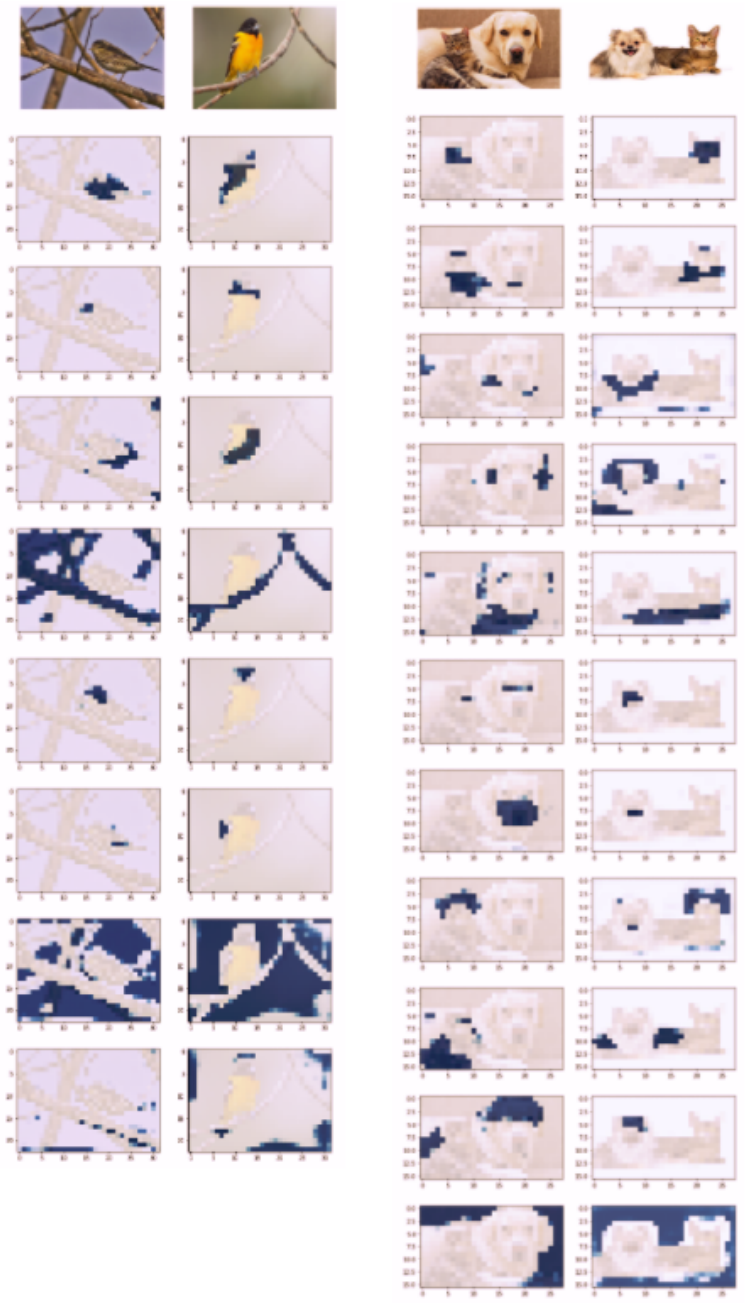

Fig. 13. More Style mask example - the last mask is the orphan mask