City Research Online

City, University of London Institutional Repository

Network of Steel: Neural Font Style Transfer from Heavy Metal to Corporate Logos

Aram Ter-Sarkisov

City, University of London

Artificial Intelligence Research Centre, CitAI

alex.ter-sarkisov@city.ac.uk

Abstract: We introduce a method for transferring style from the logos of heavy metal bands onto corporate logos using a VGG16 network. We establish the contribution of different layers and loss coefficients to the learning of style, minimization of artefacts and maintenance of readability of corporate logos. We find layers and loss coefficients that produce a good tradeoff between heavy metal style and corporate logo readability. This is the first step both towards sparse font style transfer and corporate logo decoration using generative networks. Heavy metal and corporate logos are very different artistically, in the way they emphasize emotions and readability, therefore training a model to fuse the two is an interesting problem.

1 INTRODUCTION

Recently there has been a large number of applications of convolutional neural networks (ConvNets) to neural style transfer. VGG16 (Simonyan and Zisserman, 2014) was used to extract features from both content and style images (Gatys et al., 2016) to transfer style onto a randomly created image or the content image. This approach was improved in (Zhang and Dana, 2017) by adding upsampling layers and making the network fully convolutional. A number of generative adversarial networks, GANs (Goodfellow et al., 2014) were developed and successfully applied to the neural style transfer for images and videos, such as CycleGANs (Zhu et al., 2017), Pix2pix (Isola et al., 2017), pose-guided GANs (Ma et al., 2017).

Font neural style transfer is an area of neural style transfer that is concerned with the transfer and generation of font styles. In (Azadi et al., 2018) GAN was developed that synthesizes unseen glyphs (characters) given the previously observed ones in a particular decorative style. In (Yang et al., 2019) GANs are trained to transfer style (fire, water, smoke) to glyphs to create an artistic representation. GlyphGAN (Hayashi et al., 2019) was recently developed for generation of glyphs in a required style. Neural font transfer for logo generation (Atarsaikhan et al., 2018) uses a framework similar to (Gatys et al., 2016), i.e.

minimizes distance to the style and content images by extracting features from ConvNet (VGG16) layers.

In this publication we would like to extend these findings to a sparse case of logo style transfer: from the style image, logo of a heavy metal band we want to extract only foreground font ignoring the background. Content images are corporate logos. To the best of our knowledge this is the first attempt to train such a model. In Section 2 we introduce Network of Steel that learns to transfer heavy metal logo style while maintaining corporate logo structure, Section 3 presents the main results of the experiments, Section 4 concludes.

2 OUR APPROACH

We introduce Network of Steel that learns to transfer the style from the heavy metal logo while maintaining the structure of the corporate logo. We compare two models, one based solely on VGG16 (Gatys et al., 2016) and the other on Multistyle Generative Network (Zhang and Dana, 2017). The advantage of the former is that it does not require a large dataset for training; instead, only one content and one style image are used.

2.1 Heavy metal and corporate logo style

In this publication we only concern ourselves with the style of band logos, leaving out the style of album covers, which is an entirely different area. Logos of heavy metal bands are usually carefully designed in order to convey a certain group of emotions or messages, usually those of fear, despair, aggression, alertness, eeriness, mystery. These logos are often a true work of art. Several features stand out in many logos: the first and the last glyphs are more prominent, often elongated and symmetric around the center of the logo, most glyphs are decorated in the same style, e.g. Megadeth logo glyphs have sharpened kinks at the edges (arms), the first glyph (*M*) and the last glyph (*h*) are about twice the size of other glyphs, with extrusions and slanted bottom kinks. The logo is also symmetric around the horizontal and vertical axes, see Figure 1a.

On the other hand, corporate logos are often barely distinguishable from plain text. Their design, although often expensive in development, tends to be functional, vapid and boring, with an emphasis on readability and recognizability, see Figure 1b for Microsoft logo. This publications intends to bridge the gap between the two by transferring the style from the heavy metal band (Megadeth) to a corporate logo (Microsoft).

Heavy metal band logos are an example of sparse style in a sense that we only want to learn and transfer font (glyph) features keeping the corporate logo's white background. This often leads to the creation of a large number of artefacts, such as color pixels.

2.2 VGG16 and MSG Net

We compare two models, VGG16 (Simonyan and Zisserman, 2014) used by (Gatys et al., 2016) to transfer style and multi-style generative network (Zhang and Dana, 2017), which uses Siamese network to extract features from the style image, fully convolutional network (FCN) to extract features from the content image and co-match layer to find correlation. VGG16 is presented in Figure 2 In this publication we refer to the relevant `Conv` (convolution) layer using the style adapted in most deep learning publications and code,`convi_j` , where `i` refers to the block in VGG16 and `j` to the `Conv` layer in that block. Block in VGG16 is an informal, but very useful term for our purpose. The first two blocks have 2 `Conv` layers each, with 64 and 128 feature maps, equipped with `ReLU` and

`MaxPool` layers. The next three blocks have 3 `Conv` layers each, also followed by `ReLU` and `MaxPool` layers, with 256, 512 and 512 feature maps.

2.3 Network of Steel

Following the framework of (Gatys et al., 2016), we use exactly one style image, one content image and one synthesized image that is updated every iteration of the algorithm. Content and style features are extracted once before the start of the algorithm, and features extracted from the synthesized image are compared to them every iteration to obtain the loss value, backpropagated it and update the synthesized image.

Our model, which we refer to as Network of Steel, is VGG16 without the classifier layers (`fc6`,`fc7`). Most generative networks use one last `ReLU` layer from every block for style features extraction and the second `ReLU` from the fourth block for content loss (Gatys et al., 2016). Our first contribution is the use of `Conv` layers for feature extraction and loss computation, because `ReLU` layers produce multiple artefacts. Our second contribution is the use of coarse features from the first block (`conv1_1`, `conv1_2`).

We show that extracting coarse features from the style image minimizes artefacts in the stylized logo and it is sufficient to use only two deep layers in addition to a coarse layer to transfer style without distorting the corporate logo or creating many artefacts. Finally, our third contribution are the layerwise loss weights that determine the contribution of the layer to the style loss. We show that VGG16 outperforms MSG Net for style transfer.

The specific challenge that Network of Steel faces is the sparse distribution of style features: the network only needs to learn to transfer font features (shapes, ornaments, glyph sizes and ratio, overall symmetry) to the synthesized image and keep the background neutral.

To find the content loss, we use the same layer as in (Gatys et al., 2016), `conv4_2`, and minimize mean squared error function between the features of the content and the synthesized image. Each layerwise style loss is multiplied by the predefined loss coefficient; if the coefficient is different from 0, we refer to the corresponding layer as an *active* layer:

$$\mathrm{L}_{Total} = \mathrm{L}_{Content} + \sum_{l=0}^{L} c_l E^l \qquad (1)$$

(a) Megadeth logo, black and white   (b) Microsoft logo, black and white
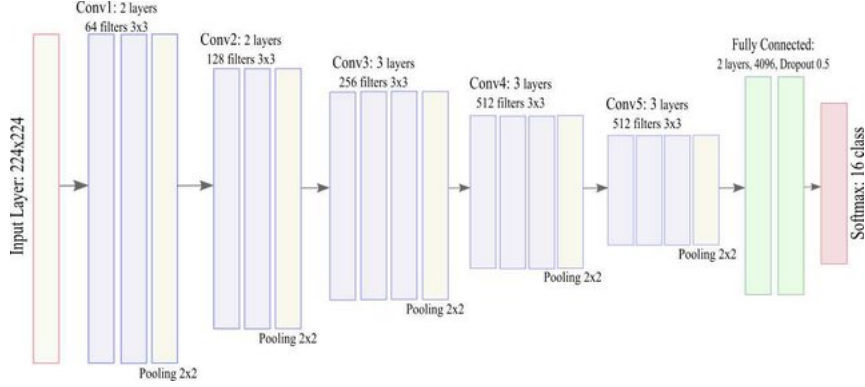
Figure 1: Examples of heavy metal and corporate logos

Figure 2: VGG16 architecture. There are in total five blocks, the first two blocks have two `Conv` layers, each followed by `ReLU` and `MaxPool` layers, the last three have three `Conv` layers, each followed by `ReLU` and `MaxPool` layers. Image taken from (Das et al., 2018).

Coefficients c_l are layerwise loss coefficients specified in Section 3 and Appendix. For example, for layer `conv3_3` layer loss coefficient is c_{3_3}. For style and transferred image we compute correlation matrices $\mathbf{A^l}$ and $\mathbf{G^l}$ for every active layer. Each element of $\mathbf{G^l}$, G_{ij}^l is a dot-product of vectorized feature maps in that layer:

$$G_{ij}^l = \mathbf{f_i^T} \cdot \mathbf{f_j} \tag{2}$$

This array is known as Gram matrix. Distance between Gram matrices for the style and synthesized images, $\mathbf{A^l}$ and $\mathbf{G^l}$ is a contribution to the style loss, and it is measured using mean squared error:

$$E^l = \frac{1}{4H^2W^2} \sum_{i,j}^{C^2} (A_{ij}^l - G_{ij}^l)^2 \tag{3}$$

Equation 1 is different from the total loss equation in (Gatys et al., 2016) because we only use layer-specific style loss coefficients, and do not use the style loss coefficient. Here H is the height of the feature map, L is the length of the feature maps and C is the number of channels/feature maps in layer l. We set most layerwise coefficients to 0, and establish which active layers contribute most to the creation of a readable corporate logo with the best heavy metal style features and least artefacts, which is the main contribution of this publication.

3 EXPERIMENTS

We compare the results of Network of Steel, MSG Net and VGG16 as implemented in the GitHub repository https://github.com/rrmina/neural-style-pytorch. VGG16 uses `Conv` layers `conv1_2`, `conv2_2`, `conv3_3`, `conv4_3`, `conv5_3`.

Weights in VGG16 which extracts features from the synthesized image are frozen for the full duration of the algorithm, and only partial derivatives for the pixels $\mathbf{x}$ in the synthesized image are computed and updated: $\frac{\partial L_{Total}}{\partial \mathbf{x}} \neq 0$.

Network of Steel is always initialized with the content image and pretrained VGG16 weights. Only specified layers contribute to the style loss. For training we use Adam optimizer with a learning rate of 1.0 and regularization constant of 0.001. We use the same MSG-Net hyperparameters as in the GitHub project: https://github.com/zhanghang1989/MSG-Net.

We select three values for layerwise coefficients: 0, 20, 200, 2000. If the coefficient is set to 0, the style loss from this layer is not computed and ignored during training. Only losses from the active layers contribute to the total loss. To test our ideas, we tried three different groups of layers to compute style loss:

1. Two layers, the fist one is always `conv1_2`, the other layer is the last `Conv` layer from one of the last three blocks.

2. Three layers, the fist one is always `conv1_2`, the other two layers are the two last `Conv` layers from one of the last three blocks.

3. Two blocks, the fist one is always the first block of VGG16, the other one is one of the last three blocks.

Obviously far more combinations of layers and loss coefficients can be tested, but empirically we determined that adding more layers and increasing loss coefficients leads to the deterioration of the network's performance, so these results are not presented.

3.1 Baseline VGG16 model

Results for the generative network from (Gatys et al., 2016) are presented in Table 1. In the original publication and code the last `ReLU` layer of each block is used with different weights, but most open-source implementations use total style loss of 1e2 and layerwise loss coefficient of 0.2. We use a higher style loss weight $= 1e5$. In addition to 0.2 we try coefficients of 0.02 and 0.002. For the lowest loss coefficent, the model displays artefacts (color pixels). For higher coefficients it distorts the font and makes the logo unreadable. This demonstrates that the selection of layers for loss computation alone is not enough for sparse style transfer, and adjusting loss coefficients affects the presence of artefacts and degree of style transfer.

3.2 Two style layers

Content features include the structure (text) of the corporate logo, but learning these features always leads to a large number of artefacts in the transferred image. To minimize these artefacts, the second `conv` layer from the first block, `conv1_2` is used. This coarse layer is responsible for the learning of color features from the style logo, which transfers the background features (white color), but does not learn font features like the edgy shapes of the glyphs in Megadeth logo. Using only one other layer from the last three blocks increases both the effect of style glyph features, including the elongation of the first and the last glyphs (elongated bottom of the front 'leg' of *M* and *t*, as particularly obvious in Tables 3 and 4, and the presence of the artefacts: font decay, colored pixels in the background, wrong style features (like a vertical 'extension' to glyph 'o' in Table 4).

Increasing loss coefficient for the deep layers always leads to the transfer of more subtle features, like the more 'rectangular' shape of the intermediate glyphs (all glyphs except the first and the last one). More sophisticated style features, like small horizontal extrusions on top of 'legs' of *M* and *t* remain a challenge with two style layers. Either the model evolves small black vertical 'blob', like in Table 2, or merges this extrusion with the dot over *i*, the next glyph. For the same glyph *M*, the elongated bottom of the front leg is a lesser challenge, (see Table 4 with `conv3_3`, to a lesser extent this is achieved in Table 3 with `conv4_3`).

It is noticeable that the last layer in the network, `conv5_3` contributes very little to the style transfer. We explain this by the size of the layer and the size of the loss it generates. `conv4_3` and `conv3_3` have either the same (512) or fewer (256) number of maps, but they are larger. Convergence of the networks with the largest coefficients, `conv1_2` = `conv5_3` = `conv4_3` = `conv3_3`=2000, is shown in Figure 3a. For the same size of the loss coefficient, `conv3_3`'s contribution to the style loss is much higher than of the other layers.

3.3 Three style layers

The results for the fifth block are very similar to the previous case with two style layers: the network fails to learn any significant style features. Same is true for the fourth block, only when the largest loss coefficient is used it manages to evolve some style features, like the elongation of the last glyph. The third block learns the style very aggressively, and for c_{3_2}=c_{3_3}=200 most style features are learnt, including the horizontal kinks on top of the first glyph, but at the cost of overall deterioration of the logo quality, see Table 7.

This is reflected in the convergence of the networks, see Figure 3b. Despite some progress compared to the networks with two layers, there seems to be a lack of good tradeoff between learning of style and maintaining the structure of the corporate logo. This is reflected in the addition of another coarse layer in the next subsection.

3.4 Two style blocks

With two blocks, we use all the `Conv` layers in each block. Empirically we established that adding even more layers or whole blocks either does not improve the result, or the logo readability starts to deteriorate.

To find a better tradeoff between readability and style transfer, we add another coarse layer, `conv1_1`, so the total number of active layers increased to five, same as in (Gatys et al., 2016). This effect explains why some results have fewer style features than in the previous subsection, despite adding another deep layer.

The overall result for the fifth block in Table 8 is still quite weak, but with the largest loss coefficients for all layers it manages to transfer some style for the first and the last glyphs, and the intermediary glyphs, with very few artefacts. The third block contributes much more to the style loss, so the best results is achieved for $c_{1\_1}=c_{1\_2}=2000$ and $c_{3\_1}=c_{3\_2}=c_{3\_3}=20$, but further increase of style loss coefficients leads to the deterioration of the transferred logo. Nevertheless, for the largest loss coefficients the network almost correctly evolves the first and the last glyphs at the cost of adding few background artefacts and slight deterioration of the readability of the synthesized logo.

The best results that balance the heavy metal style and corporate logo readability were obtained using the fourth block with the style loss coefficients of 200 and $c_{1\_1}=c_{1\_2}=2000$ in Table 9: the model evolved most of the challenging features of the metal logo without a heavy presence of artefacts. They are an improvement over the results in Table 6 in most ways, which proves that adding both a deep style layer and a coarse layer to maintain the content font structure improves the final result. This could be seen in Figure 3c: the third block generates more style loss than the other two blocks, but produces an overall worse result than the fourth block that manages to maintain better readability.

3.5 MSG Net

MSG Net was introduced in (Zhang and Dana, 2017).We finetuned it to our data that we scraped from the internet: 19 corporate logos (content) and 11 heavy metal logos (style). Style loss hyperparameter was set to 10000, content loss hyperparameter to 1, learning rate to 1.0.

Although MSG Net is more advanced than plain VGG16: it has a fully convolutional architecture, learns weights to evolve an image with the transferred style and has more loss functions, it performs worse than Network of Steel in terms of sparse style transfer, as it does not transfer any font style from heavy metal

logos onto the font in the corporate logos at all. MSG-Net manages to evolve some small elements around the glyphs, that are barely noticeable.

4 CONCLUSIONS

Sparse style transfer requires an approach different to that of other neural style transfer problems, due to a large number of artefacts, merging and distortion of elements of the style and font.

In this publication we introduced Network of Steel for sparse style transfer from heavy metal band to corporate logos. We showed that in order to synthesize a readable logo with heavy metal style elements, instead of using layers from all blocks of VGG16, only one or two coarse layers and two or three deep layers are enough. Our future work includes the following challenges:

1. Train a separate network for loss coefficients,

2. Build a large database for training Networks of Steel for different heavy metal styles and corporate logos,

3. Design accuracy metrics applicable to this problem to enable visual comparison,

4. In this paper we only used a white background for heavy metal logos, which causes a lot of artefacts. In the future we will use different, more challenging backgrounds, like album covers.

We showed that `conv1_2` is essential to maintaining artefact-free background and layers from the third block in VGG16 learn style faster than deeper layers like `conv5_3` and `conv4_3`. Our approach is simple and more robust than (Gatys et al., 2016) and (Zhang and Dana, 2017) for sparse style transfer. The whole deep fourth block (`conv4_1`, `conv4_2`, `conv4_3`) with loss coefficients of 200 and two coarse layers (`conv1_1` and `conv1_2`) with loss coefficients of 2000 produce the best tradeoff between heavy metal style and the readability of the corporate logo.

REFERENCES

Atarsaikhan, G., Iwana, B. K., and Uchida, S. (2018). Contained neural style transfer for decorated logo generation. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 317–322. IEEE.

Azadi, S., Fisher, M., Kim, V. G., Wang, Z., Shechtman, E., and Darrell, T. (2018). Multi-content gan for few-shot

font style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7564–7573.

Das, A., Roy, S., Bhattacharya, U., and Parui, S. K. (2018). Document image classification with intra-domain transfer learning and stacked generalization of deep convolutional neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3180–3185. IEEE.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423.

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.

Hayashi, H., Abe, K., and Uchida, S. (2019). Glyph-gan: Style-consistent font generation based on generative adversarial networks. *arXiv preprint arXiv:1905.12502*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., and Van Gool, L. (2017). Pose guided person image generation. In *Advances in Neural Information Processing Systems*, pages 406–416.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Yang, S., Wang, Z., Wang, Z., Xu, N., Liu, J., and Guo, Z. (2019). Controllable artistic text style transfer via shape-matching gan. *arXiv preprint arXiv:1905.01354*.

Zhang, H. and Dana, K. (2017). Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

APPENDIX

Here we present some of the results that demonstrate the effect of different layer loss coefficients for logo style transfer. In each experiment, all excluded (inactive) layer loss coefficients are set to 0. All experiments were run for 50000 iterations with the same learning rate and content loss weight of 1, network gradients switched off. Due to the differences in the architecture and training, MSG-Net was evaluated on a number of heavy metal and corporate logos, the best results presented in 11.

Table 1: Results for VGG16 model, layers conv1_2, conv2_2, conv3_3, conv4_3, conv5_3, as defined in (Gatys et al., 2016)

$c_{1\_2}=c_{2\_2}=c_{3\_3}=c_{4\_3}=c_{5\_3}=20$	$c_{1\_2}=c_{2\_2}=c_{3\_3}=c_{4\_3}=c_{5\_3}=200$	$c_{1\_2}=c_{2\_2}=c_{3\_3}=c_{4\_3}=c_{5\_3}=2000$

Table 2: Results for layers conv1_2 and conv5_3
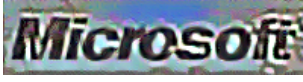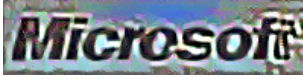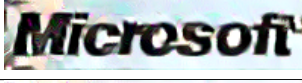
	$c_{5\_3}=20$	$c_{5\_3}=200$	$c_{5\_3}=2000$
$c_{1\_2}=0$			
$c_{1\_2}=20$			
$c_{1\_2}=200$			
$c_{1\_2}=2000$			

Table 3: Results for layers conv1_2 and conv4_3
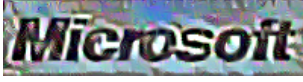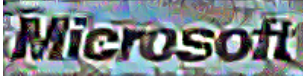
	$c_{4\_3}=20$	$c_{4\_3}=200$	$c_{4\_3}=2000$
$c_{1\_2}=0$			
$c_{1\_2}=20$			
$c_{1\_2}=200$			
$c_{1\_2}=2000$			

Table 4: Results for layers conv1_2 and conv3_3

	$c_{3\_3}=20$	$c_{3\_3}=200$	$c_{3\_3}=2000$
$c_{1\_2}=0$			
$c_{1\_2}=20$			
$c_{1\_2}=200$			
$c_{1\_2}=2000$			

Table 5: Results for layers `conv1_2` and `conv5_2`, `conv5_3`

	$c_{5\_2}=c_{5\_3}=20$	$c_{5\_2}=c_{5\_3}=200$	$c_{5\_2}=c_{5\_3}=2000$
$c_{1\_2}=0$			
$c_{1\_2}=20$			
$c_{1\_2}=200$			
$c_{1\_2}=2000$			

Table 6: Results for layers `conv1_2` and `conv4_2`, `conv4_3`

	$c_{4\_2}=c_{4\_3}=20$	$c_{4\_2}=c_{4\_3}=200$	$c_{4\_2}=c_{4\_3}=2000$
$c_{1\_2}=0$			
$c_{1\_2}=20$			
$c_{1\_2}=200$			
$c_{1\_2}=2000$			

Table 7: Results for layers `conv1_2` and `conv3_2`, `conv3_3`

	$c_{3\_2}=c_{3\_3}=20$	$c_{3\_2}=c_{3\_3}=200$	$c_{3\_2}=c_{3\_3}=2000$
$c_{1\_2}=0$			
$c_{1\_2}=20$			
$c_{1\_2}=200$			
$c_{1\_2}=2000$			
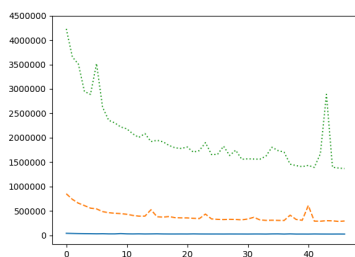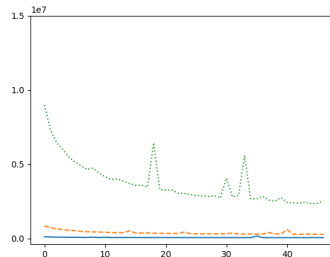
(a) Two style layers   (b) Three style layers   (c) Two style blocks
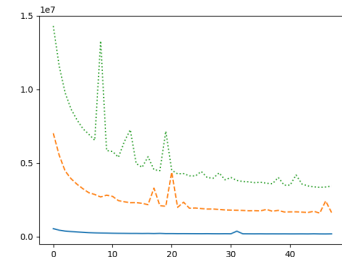
Figure 3: Style loss for each network with all loss coefficients = 2000 plotted against thousands of iterations. Continuous curve: fifth block layers, dashed curve: fourth block layers, dotted curve: third block layers.

Table 8: Results for the first and the fifth block: `conv1_1 conv1_2`, and `conv5_1, conv5_2, conv5_3`
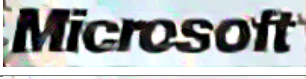
	$c_{5\_1}=c_{5\_2}=c_{5\_3}=20$	$c_{5\_1}=c_{5\_2}=c_{5\_3}=200$	$c_{5\_1}=c_{5\_2}=c_{5\_3}=2000$
$c_{1\_1}=c_{1\_2}=0$			
$c_{1\_1}=c_{1\_2}=20$			
$c_{1\_1}=c_{1\_2}=200$			
$c_{1\_1}=c_{1\_2}=2000$			

Table 9: Results for the first and the fourth block: `conv1_1 conv1_2`, and `conv4_1, conv4_2, conv4_3`

	$c_{4\_1}=c_{4\_2}=c_{4\_3}=20$	$c_{4\_1}=c_{4\_2}=c_{4\_3}=200$	$c_{4\_1}=c_{4\_2}=c_{4\_3}=2000$
$c_{1\_1}=c_{1\_2}=0$			
$c_{1\_1}=c_{1\_2}=20$			
$c_{1\_1}=c_{1\_2}=200$			
$c_{1\_1}=c_{1\_2}=2000$			

Table 10: Results for the first and the third block: `conv1_1 conv1_2`, and `conv3_1, conv3_2, conv3_3`

	$c_{3\_1}=c_{3\_2}=c_{3\_3}=20$	$c_{3\_1}=c_{3\_2}=c_{3\_3}=200$	$c_{3\_1}=c_{3\_2}=c_{3\_3}=2000$
$c_{1\_1}=c_{1\_2}=0$			
$c_{1\_1}=c_{1\_2}=20$			
$c_{1\_1}=c_{1\_2}=200$			
$c_{1\_1}=c_{1\_2}=2000$			

Table 11: Results for MSG-Net model

	amazon	IKEA	Microsoft
MEGADETH	amazon	IKEA	Microsoft
MANOWAR	amazon	IKEA	Microsoft
MAYHEM	amazon	IKEA	Microsoft
DOOM	amazon	IKEA	Microsoft
BENEDICTION	amazon	IKEA	Microsoft