– Supplementary Material –
Neural Inverse Knitting: From Images to Manufacturing Instructions

Contents

Additional Resources

Additional knitting-related resources (the dataset, code and overview videos of the machine knitting process) can be found on our project page:
`http://deepknitting.csail.mit.edu/`

The **`Refiner`** Network

Our refinement network translates real images into regular images that look similar to synthetic images. Its implementation is similar to `Img2prog`, except that it outputs the same resolution image as input, of which illustration is shown in Figure A.1.

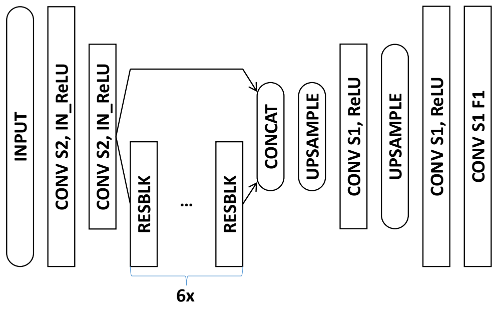

Figure A.1. The illustration of the `Refiner` network architecture, where S$\#N$ denotes the stride size of $\#N$, `IN_ReLU` indicates the Instance normalization followed by ReLU, `Resblk` is the residual block that consists of ConvS1-ReLU-ConvS1 with shortcut connection (He et al., 2016), Upsample is the nearest neighbor upsampling with the factor $2\times$, F is the output channel dimension. If not mentioned, the default parameters for all the convolutions are the stride size of 2, $F = 64$, and the 3×3 kernel size.

Loss Balancing Parameters

When learning our full architecture with both `Refiner` and `Img2prog`, we have three different losses: the cross-entropy loss $\mathcal{L}_{CE}$, the perceptual loss $\mathcal{L}_{\text{Perc}}$, and the Patch-GAN loss.

Our combined loss is the weighted sum

$$\mathcal{L} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{Perc}}\mathcal{L}_{\text{Perc}} + \lambda_{\text{GAN}}\mathcal{L}_{\text{GAN}} \quad (1)$$

where we used the weights: $\lambda_{\text{CE}} = 3$, $\lambda_{\text{Perc}} = 0.02/(128)^2$ and $\lambda_{\text{GAN}} = 0.2$. The losses $\mathcal{L}_{\text{Perc}}$ and λ_{GAN} are measured on the output of `Refiner`, while the loss λ_{CE} is measured on `Img2prog`.

The perceptual loss (Johnson et al., 2016) consists of the feature matching loss and style loss (using the gram matrix). If not mentioned here, we follow the implementation details of (Johnson et al., 2016), where VGG-16 (Simonyan & Zisserman, 2014) is used for feature extraction, after replacing max-pooling operations with average-pooling. The feature matching part is done using the `pool3` layer, comparing the input real image and the output of `Refiner` so as to preserve the content of the input data. For the style matching part, we use the gram matrices of the {`conv1_2`, `conv2_2`, `conv3_3`} layers with the respective relative weights {0.3, 0.5, 1.0}. The measured style loss is between the synthetic image and the output of `Refiner`.

For $\mathcal{L}_{\text{GAN}}$ and the loss for the discriminator, the least-square Patch-GAN loss (Zhu et al., 2017) is used. We used $\{-1, 1\}$ for the regression labels for respective fake and real samples insted of the label $\{0, 1\}$ used in (Zhu et al., 2017).

For training, we normalize the loss λ_{CE} to be balanced according to the data ratio of a batch. Specifically, for example, suppose a batch consisting of 2 real and 4 synthetic samples, respectively. Then, we inversely weighted the respective cross entropy losses for real and synthetic data by the weights of $\frac{4}{6}$ and $\frac{2}{6}$, so that the effects from the losses are balanced. This encourages the best performance to be expected at near $\alpha = 0.5$ within a batch.

Data Augmentation

We use multiple types of data augmentation to notably increase the diversity of yarn colors, lighting conditions, yarn tension, and scale:

- **Global Crop Perturbation**: we add random noise to the location of the crop borders for the real data images, and crop on-the-fly during training; the noise intensity is chosen such that each border can shift at most by half of one stitch;
- **Local Warping**: we randomly warp the input images locally using non-linear warping with linear RBF kernels on a sparse grid. We use one kernel per instruction

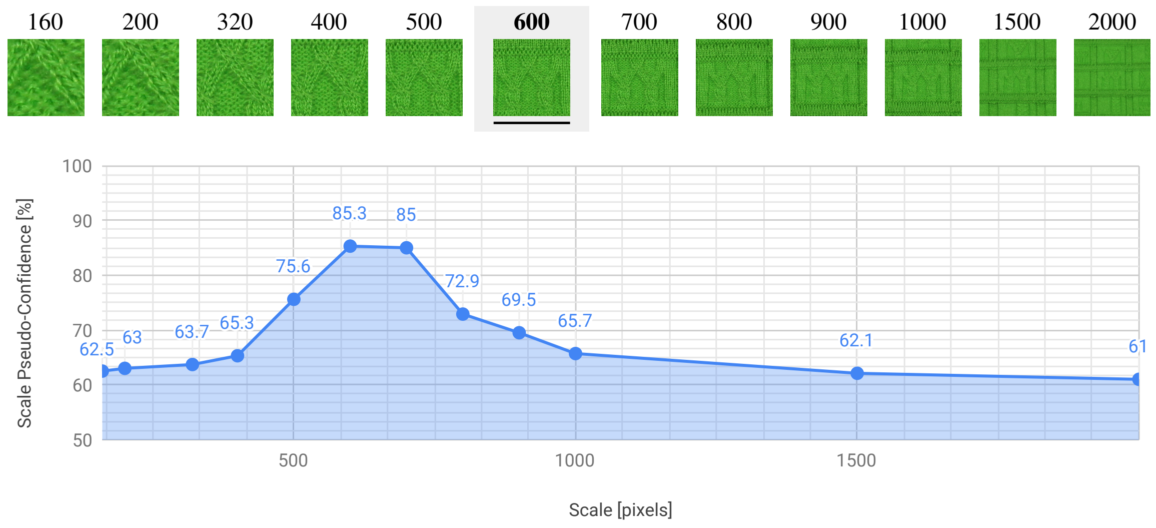

Figure A.2. Scale identification experiment. **Top row**: cropped input image at corresponding scales with the correct pixel scale in bold with a light-gray background. **Plot**: pseudo-confidence curve showing a peak at the correct pixel scale (600).

and the shift noise is a 0-centered gaussian with σ being $1/5$ of the default instruction extent in image space (i.e. $\sigma = 8/5$);

- **Intensity augmentation**: we randomly pick a single color channel and use it as a mono-channel input, so that it provides diverse spectral characteristics. Also note that, in order to enhance the intensity scale invariance, we apply instance normalization (Ulyanov et al., 2016) for the upfront convolution layers of our encoder network.

Pattern scale identification

Our base system assumes that the input image is taken at a specific zoom level designed for our dataset, which is likely not going to be true for a random image. We currently assume this to be solved by the user given proper visual feedback (i.e., the user would see the pattern in real-time as they scan their pattern of interest with a mobile phone).

Here, we investigate the potential of automatically discovering the scale of the pattern. Our base idea is to evaluate the confidence of the output instruction map for different candidate scales and to choose the one with highest confidence. Although the softmax output cannot directly be considered as a valid probability distribution, it can serve as an approximation, which can be calibrated for (Guo et al., 2017). As a proof of concept, we take a full 5×5 pattern image from our dataset and crop its center at different scales from 160 pixels to 2000 pixels of width. We then measure the output of the network and compute a *scale pseudo-confidence* as the average over pixels of the maximum softmax component.

In Figure A.2, we show a sample image with crops at various scales, together with the corresponding uncalibrated pseudo-confidence measure, which peaks at around 600 pixels scale. Coincidentally, this corresponds to the scale of our ground truth crops for that image.

This suggests two potential scenarios: (1) the user takes a much larger image and then that pattern image gets analysed offline to figure out the correct scale to work at using a similar procedure, and then generates a full output by using a tiling of crops at the detected scale, or (2) an interactive system could provide scale information and suggest the user to get closer to (or farther from) the target depending on the confidence gradient.

Data post-processing

As mentioned in the main paper, our framework does not enforce hard constraint on the output semantics. This implies that some outputs may not be machine-knittable as-is.

More precisely, the output of our network may contain invalid instructions pairs or a lack thereof. We remedy to these conflicts by relaxing the conflicting instruction, which happens in only two cases:

1. Unpaired CROSS instructions – we reduce such instructions into their corresponding MOVE variants (since CROSS are MOVEs with relative scheduling), and

2. CROSS pairs with conflicting schedules (e.g., both pair sides have same priority, or instructions within a pair's side having different priorities) – in this case, we randomly pick a valid schedule (note that its impact is only local).

Table A.1. **Performance comparison with larger scene parsing network from (Zhou et al., 2018).** (d2) uses pre-training on ImageNet (Russakovsky et al., 2015) and a much larger number of parameters (1.4M v.s. 51.4M).

	Method	Accuracy (%)		Perceptual		# Parameters
		Full	FG	SSIM	PSNR [dB]	(in Millions)
(d1)	Refiner + img2prog++ ($\alpha = 1/2$)	94.01	80.30	0.899	23.56	1.4
(d2)	Large Scene Parsing w/ pre-training	**94.95**	**83.46**	**0.908**	**24.58**	51.4

This is sufficient to allow knitting on the machine. Note that STACK are semantically *supposed* to appear with a MOVE, but they dont prevent knitting since their operations lead to the same as KNIT when unpaired, and thus do not require any specific post-processing.

Additional quantitative results

The focus of the experiments in the main paper was on assessing specific trends such as the impact of the dataset size, or the different behaviours of baseline networks, the impact of mixing data types and the ratios of these.

As can be noted, we used a standard (residual) architecture and tried to avoid over-engineering our network or its parameters. However, we provide here results that show that we can obviously still do better by using more complex and larger networks, to the detriment of having to train for a longer time and resulting in a much larger model size.

In our baseline, we compared with a sample architecture from (Zhou et al., 2018), which we made small enough to compare with our baseline `Img2prog` implementation. Furthermore, our baseline implementations were all trained from scratch and did not make use of pre-training on any other dataset.

Here, we provide results for a much larger variant of that network, which we name *Large Scene Parsing*, and makes use of pre-training on ImageNet (Russakovsky et al., 2015). The quantitative comparison is provided in Table A.1, which shows that we can achieve even better accuracy than our best current results using our `Refiner`+`Img2prog`++ combination. However, note that this comes with a much larger model size: ours has $1.4M$ parameters[1], whereas *Large Scene Parsing* has $51.4M$. Furthermore, this requires pre-training on ImageNet with millions of images (compared to our model working with a few thousands only).

Additional qualitative results

We present additional qualitative results obtained from several networks in Figure A.3.

[1]M for Million

Proof of Theorem 1

We first describe the necessary definitions and lemmas to prove Theorem 1. We need a general way to measure the discrepancy between two distributions, which we borrow from the definition of discrepancy suggested by (Mansour et al., 2009).

Definition 1 (Discrepancy (Mansour et al., 2009))**.** *Let $\mathcal{H}$ be a class of functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. The discrepancy between two distribution $\mathcal{D}_1$ and $\mathcal{D}_2$ over $\mathcal{X}$ is defined as*

$$\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_1, \mathcal{D}_2) = \max_{h,h' \in \mathcal{H}} |\mathcal{L}_{\mathcal{D}_1}(h, h') - \mathcal{L}_{\mathcal{D}_2}(h, h')| \,. \quad (2)$$

The discrepancy is symmetric and satisfies the triangle inequality, regardless of any loss function. This can be used to compare distributions for general tasks even including regression.

The following lemma is the extension of Lemma 4 in (Ben-David et al., 2010) to be generalized by the above discrepancy.

Lemma 1. *Let h be a hypothesis in class $\mathcal{H}$, and assume that $\mathcal{L}$ is symmetric and obeys the triangle inequality. Then*

$$|\mathcal{L}_\alpha(h, y) - \mathcal{L}_T(h, y)| \leq \alpha \left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right), \quad (3)$$

where $\lambda{=}\mathcal{L}_S(h^*, y){+}\mathcal{L}_T(h^*, y)$, and the ideal joint hypothesis h^* is defined as $h^*{=}\arg\min_{h\in\mathcal{H}} \mathcal{L}_S(h, y){+}\mathcal{L}_T(h, y)$.

Proof. The proof is based on the triangle inequality of $\mathcal{L}$, and the last inequality follows the definition of the discrepancy.

$$\begin{aligned}
&|\mathcal{L}_\alpha(h, y) - \mathcal{L}_T(h, y)| \\
=&\alpha|\mathcal{L}_S(h, y) - \mathcal{L}_T(h, y)| \\
=&\alpha\,|\mathcal{L}_S(h, y) - \mathcal{L}_S(h^*, h) + \mathcal{L}_S(h^*, h) \\
&\ - \mathcal{L}_T(h^*, h) + \mathcal{L}_T(h^*, h) - \mathcal{L}_T(h, y)\,| \\
\leq&\alpha\big|\,|\mathcal{L}_S(h, y) - \mathcal{L}_S(h^*, h)| + \\
&\ |\mathcal{L}_S(h^*, h) - \mathcal{L}_T(h^*, h)| + |\mathcal{L}_T(h^*, h) - \mathcal{L}_T(h, y)|\,\big| \\
\leq&\alpha\big|\mathcal{L}_S(h^*, y){+}|\mathcal{L}_S(h^*, h){-}\mathcal{L}_T(h^*, h)|{+}\mathcal{L}_T(h^*, y)\big| \\
\leq&\alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right). \qquad (4)
\end{aligned}$$

We conclude the proof. □

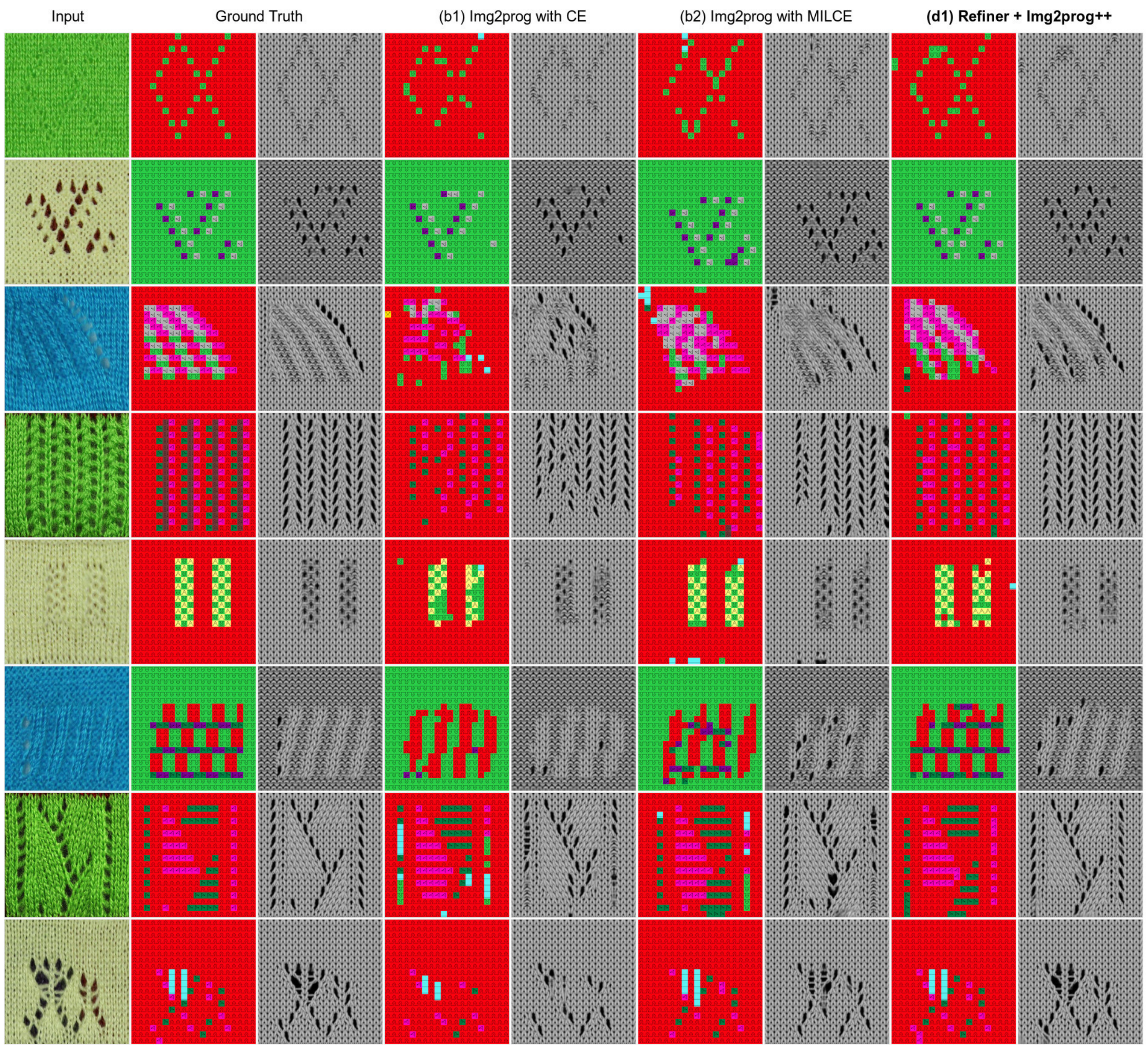

Figure A.3. Additional comparisons of instructions predicted by different version of our method. We present the predicted instructions as well as a corresponding image from our renderer.

Many types of losses satisfy the triangle inequality, e.g., the $0-1$ loss (Ben-David et al., 2010; Crammer et al., 2008) and l_1-norm obey the triangle inequality, and l_p-norm ($p > 1$) obeys the pseudo triangle inequality (Galanti & Wolf, 2017).

Lemma 1 bounds the difference between the target loss and α-mixed loss. In order to derive the relationship between a true expected loss and its empirical loss, we rely on the following lemma.

Lemma 2 ((Ben-David et al., 2010))**.** *For a fixed hypothesis h, if a random labeled sample of size m is generated by drawing βm points from $\mathcal{D}_S$ and $(1-\beta)m$ points from $\mathcal{D}_T$, and labeling them according to y_S and y_T respectively, then for any $\delta \in (0,1)$, with probability at least $1-\delta$ (over the choice of the samples),*

$$|\hat{\mathcal{L}}_\alpha(h,y) - \mathcal{L}_\alpha(h,y)| \leq \epsilon(m,\alpha,\beta,\delta), \tag{5}$$

where $\epsilon(m,\alpha,\beta,\delta) = \sqrt{\frac{1}{2m}\left(\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}\right)\log(\frac{2}{\delta})}$.

The detail function form of ϵ will be omitted for simplicity. We can fix m, α, β, and δ when the learning task is specified, then we can treat $\epsilon(\cdot)$ as a constant.

Theorem 1. *Let $\mathcal{H}$ be a hypothesis class, and $\mathcal{S}$ be a labeled sample of size m generated by drawing βm samples from $\mathcal{D}_S$ and $(1-\beta)m$ samples from $\mathcal{D}_T$ and labeling them according to the true label y. Suppose $\mathcal{L}$ is symmetric and obeys the triangle inequality. Let $\hat{h} \in \mathcal{H}$ be the empirical minimizer of $\hat{h} = \arg\min_h \hat{\mathcal{L}}_\alpha(h, y)$ on $\mathcal{S}$ for a fixed $\alpha \in [0, 1]$, and $h^*_T = \arg\min_h \mathcal{L}_T(h, y)$ the target error minimizer. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ (over the choice of the samples), we have*

$$|\mathcal{L}_T(\hat{h}, y) - \mathcal{L}_T(h^*_T, y)| \le 2\left(\alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right) + \epsilon\right), \tag{6}$$

where $\epsilon(m, \alpha, \beta, \delta) = \sqrt{\frac{1}{2m}\left(\frac{\alpha^2}{\beta} + \frac{(1-\alpha)^2}{1-\beta}\right)\log(\frac{2}{\delta})}$, *and* $\lambda = \min_{h\in\mathcal{H}} \mathcal{L}_S(h, y) + \mathcal{L}_T(h, y)$.

Proof. We use Lemmas 1 and 2 for the bound derivation with their associated assumptions.

$$\begin{aligned}
&\mathcal{L}_T(\hat{h}, y) \\
&\le \mathcal{L}_\alpha(\hat{h}, y) + \alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right), && (7) \\
&&& \text{(By Lemma 1)} \\
&\le \hat{\mathcal{L}}_\alpha(\hat{h}, y) + \alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right) + \epsilon, && (8) \\
&&& \text{(By Lemma 2)} \\
&\le \hat{\mathcal{L}}_\alpha(h^*_T, y) + \alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right) + \epsilon, && (9) \\
&&& (\hat{h} = \arg\min_{h\in\mathcal{H}} \hat{\mathcal{L}}_\alpha(h)) \\
&\le \mathcal{L}_\alpha(h^*_T, y) + \alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right) + 2\epsilon, && (10) \\
&&& \text{(By Lemma 2)} \\
&\le \mathcal{L}_T(h^*_T, y) + 2\alpha\left(\mathrm{disc}_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) + \lambda\right) + 2\epsilon, && (11) \\
&&& \text{(By Lemma 1)}
\end{aligned}$$

which concludes the proof. □

Theorem 1 does not have unnecessary dependencies for our purpose, which are used in (Ben-David et al., 2010) such as unsupervised data and the restriction of the model type to finite VC-dimensions.

References

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Crammer, K., Kearns, M., and Wortman, J. Learning from multiple sources. *Journal of Machine Learning Research*, 9(Aug):1757–1774, 2008.

Galanti, T. and Wolf, L. A theory of output-side unsupervised domain adaptation. *arXiv:1703.01606*, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330. JMLR. org, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

Mansour, Y., Mohri, M., and Rostamizadeh, A. Domain adaptation: Learning bounds and algorithms. In *Conference on Learning Theory*, 2009.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

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

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., and Torralba, A. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 2018.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, 2017.