

# PIXCOLOR: PIXEL RECURSIVE COLORIZATION

**Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, Kevin Murphy**  
 Google Research  
 {sguada, rld, dbieber, mnorouzi, shlens, kpmurphy}@google.com

## ABSTRACT

We propose a novel approach to automatically produce multiple colorized versions of a grayscale image. Our method results from the observation that the task of automated colorization is relatively easy given a low-resolution version of the color image. We first train a conditional PixelCNN to generate a low resolution color for a given grayscale image. Then, given the generated low-resolution color image and the original grayscale image as inputs, we train a second CNN to generate a high-resolution colorization of an image. We demonstrate that our approach produces more diverse and plausible colorizations than existing methods, as judged by human raters in a "Visual Turing Test".

## 1 INTRODUCTION

Building a computer system that can automatically convert a black and white image to a plausible color image is useful for restoring old photographs, videos [34], or even assisting cartoon artists [26, 32]. From a computer vision perspective, this may appear like a straightforward image-to-image mapping problem, amenable to a convolutional neural network (CNN). We denote this by  $y = f(x)$ , where  $x$  is the input grayscale image,  $y$  is the predicted color image, and  $f$  is a CNN. This approach has been pursued in several recent papers [5, 15, 20, 41, 10, 7, 17] which leverages the fact that one may obtain unlimited labeled training pairs by converting color images to grayscale.

Removing the chromaticity from an image is a surjective operation, thus restoring color to an image is a one-to-many operation (Figure 1). We can express this ambiguity as a conditional probability model  $y \sim p(y | x)$  to capture multiple possible outputs, rather than predicting a single image (see Section 2 for review of generative models).

In this paper, we propose a new method, that employs a PixelCNN [36] probabilistic model to produce a coherent joint distribution over color images given a grayscale input. PixelCNNs have several advantages over other conditional generative models: (1) they capture dependencies between the pixels to ensure that colors are selected consistently; (2) the log-likelihood can be computed exactly and training is stable unlike other generative models.

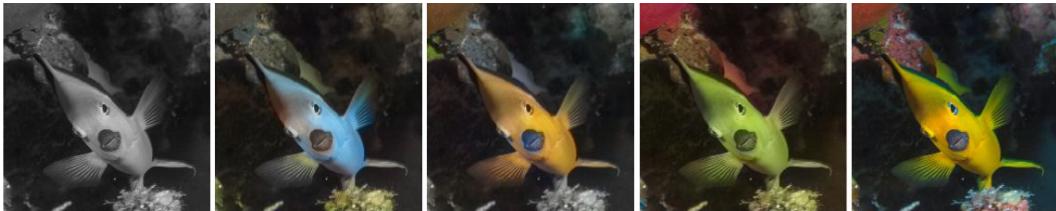


Figure 1: Grayscale on the left with three colorizations from our model and the original.

The main disadvantage of PixelCNNs, however, is that they are slow to sample from, due to their inherently sequential (autoregressive) structure. In this paper we leverage the fact that the chrominance of an image (especially as perceived by humans) is of much lower spatial frequency than the luminance. In fact, some image storage formats, such as JPEG, exploit this intuition and store the color channels at lower resolution than the intensity channel. This means that it is sufficient for the PixelCNN to predict a low resolution color image, which may be done quite quickly. We then train a

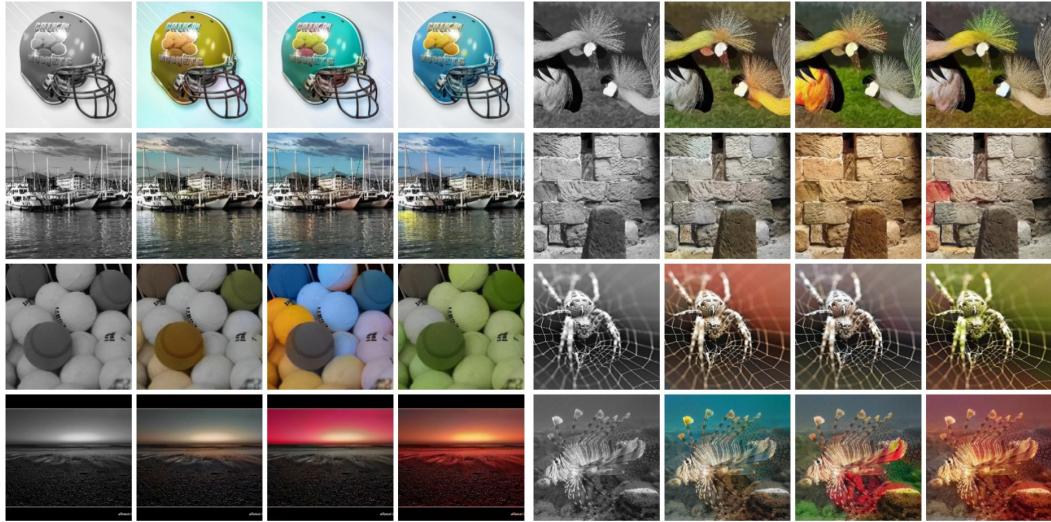


Figure 2: Diverse colorizations generated by our PixColor method. For each group of 4 images, the first is the grayscale input, and the rest are 3 samples from the model.

second CNN-based “refinement network”, which combines the predicted low resolution color image with the high resolution grayscale input to produce a high resolution color image.

Formally, our approach can be thought of as a conditional latent variable model of the form  $p(y | x) = \sum_z \delta(y = f(x, z))p(z | x)$ , where  $x$  is the input grayscale image,  $y$  is the output color image,  $z$  is the latent low-dimensional color image. The PixelCNN estimates  $p(z | x)$ , and the refinement CNN estimates  $y = f(x, z)$ . At test time, rather than summing over  $z$ ’s, we sample a few  $z$ ’s. During training, we use the ground truth low resolution color image for  $z$ , so that we can fit the two conditional models independently. See Section 3 for the details.

Our proposed method, called Pixel Recursive Colorization (PixColor), produces diverse, high quality colorizations. Figure 2 depicts some examples with high diversity. In Section 4, we describe how we quantitatively evaluate the performance of colorization using human raters. We report our results in Section 5, where we show that PixColor significantly outperforms existing methods. Section 6 concludes the paper and discusses some future directions.

## 2 RELATED WORK

Early approaches to colorization relied on some amount of human effort, either to identify a relevant source color image from which the colors could be transferred [38, 6, 13, 16, 24, 33, 28, 25, 3], or to get a rough coloring from a human annotator to serve as a set of “hints” [21, 14, 22, 26, 40, 42, 11]. More recently, there has been a surge of interest in developing fully automated solutions, which do not require human interaction (see Table 1).

Most recent methods train a CNN to map a gray input image to a single color image [5, 15, 20, 41, 10, 7]. When such models are trained with L2 or L1 loss, the colorization results often look somewhat “washed out”, since the model is encouraged to predict the average color. Some recent papers (*e.g.*, [20, 41]) discretize the color space, and use a per-pixel cross-entropy loss on the softmax outputs of a CNN, resulting in more colorful pictures, especially if rare colors are upweighted during training (*e.g.*, [41]). However, since the model predicts each pixel independently, the one-to-many nature of the task is not captured properly, *e.g.*, all of the pixels in a region cannot be constrained to have the same color.

Previous work has proposed several ways to ensure that multiple colorizations generated by a model are globally coherent. One approach is to use a conditional random field (CRF) [3], although inference in such models can be slow. A second approach is to use a CNN with multiple output “heads”, corresponding to different colorizations of an image. One can additionally train a “gating” network

Name/Ref.	Model	Color	Loss	Multi	Dataset
AICMP [3]	CRF	Lab	N/A	N	N/A
LTBC [15]	CNN	Lab	L2 + class CE	N	MIT places
LRAC [20]	CNN	Lab	CE	N	ImageNet
CIC [41]	CNN	Lab	CE	N	ImageNet
MOE [1]	MOE	YCbCr	L2	Y	ImageNet
VAE [9]	MDN + VAE	Lab	Mahal.	Y	ImageNet
Pix2Pix [17]	GAN	Lab	Adv.	N	ImageNet
GAN [2]	GAN	YUV	Adv.	N	LSUN
PixColor (this paper)	PixelCNN + CNN	YCbCr	CE + L1	Y	ImageNet

Table 1: Summary of related methods. Columns comprise name of method; reference; model type (MOE = mixture of experts, VAE = variational autoencoder, MDN = mixture density network, GAN = generative adversarial network); color space; loss (CE = cross entropy, Mahal = Mahalanobis distance, Adv = adversarial); multiple diverse outputs or not; and the dataset used to train the model. The CRF method of [3] requires that the user specify one or more training images that are similar to the input gray image. Although the CRF is capable of generating multiple solutions, [3] uses graph-cuts to produce a single MAP estimate. Similarly, although the GAN method of [17] is capable of producing multiple solutions, they report that their GAN ignores the noise, and always predicts the same answer for each input. This problem is fixed in [2] by introducing noise at multiple levels of the generator.

to select the best head for a given image. This mixture of experts (MOE) approach was used in [1] mainly for image compression, rather than colorization *per se*.

A third approach is to use a (conditional) variational autoencoder (VAE) [18] to capture dependencies amongst outputs via a low dimensional latent space. To capture the dependence on the input image, [9] proposes to use a mixture density network (MDN) to learn a mapping from a gray input image to a distribution over the latent codes, which is then converted to a color image using the VAE’s decoder. Unfortunately, this method often produces sepia toned results (Table 3).

A fourth approach is to use a (conditional) generative adversarial network (GAN) [12] to train a generative model jointly with a discriminative model. The goal of the discriminative model is detect synthesized images, while the goal of the generative model is fool the discriminator. This approach results sharp images, but [17] reports that a GAN-based colorization results underperform previous CNN approaches [41]. One of their failure modes “mode collapse” problem, whereby the resulting model correctly predicts one mode of a distribution but fails the full diversity of the data [23]. More recently, [2] have applied a slightly different GAN to colorization. Although the authors claim to avoid the mode collapse problem, it is hard to compare against previous results because the authors only employ the LSUN-bedrooms dataset for evaluation. Most papers (including ours) employ the “ctest10k” split of the ImageNet validation dataset from [20] (see Section 4 for more details).

We propose a novel approach that uses a PixelCNN [36] to produce multiple low resolution color images, which are then deterministically converted to high resolution color images using a CNN refinement network. By using multiple low resolution color “hints” to the CNN, we capture the one-to-many nature of the task and prevent the CNN from producing sepia toned outputs.

Very recently, in a concurrent submission, [29] proposed an approach which is similar to ours. However, instead of passing the output of a pixelCNN into a refinement CNN, they do the opposite, and pass the output a CNN into a pixelCNN. The visual quality and diversity of their results look good, but, unlike us, they do not perform any human evaluation, so we do not have a quantitative comparison. The primary disadvantage of their approach is that it is slow for a pixelCNN to generate high resolution images; indeed, their method only generates  $32 \times 32$  color images, which are then deterministically upscaled to  $128 \times 128$ . By contrast, our CNN refinement network learns to upscale from  $28 \times 28$  to the same size as the input, which works much better than deterministic upscaling, as we will show. We mostly focus on generating  $256 \times 256$  images, to be comparable to prior work, but we also show some non-square examples, which is important in practice, since many grayscale photos of interest are in portrait or landscape mode.

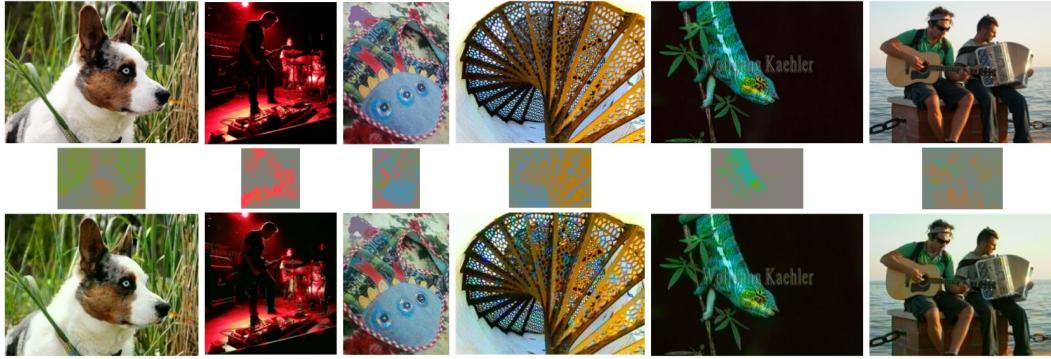


Figure 3: All you need is a few bits of color. The top row is the original color image. The middle row is the true chroma image downsampled to have smallest size 28 pixels. The bottom row is the result of bilinear upsampling the middle row, and combining with the original grayscale image.

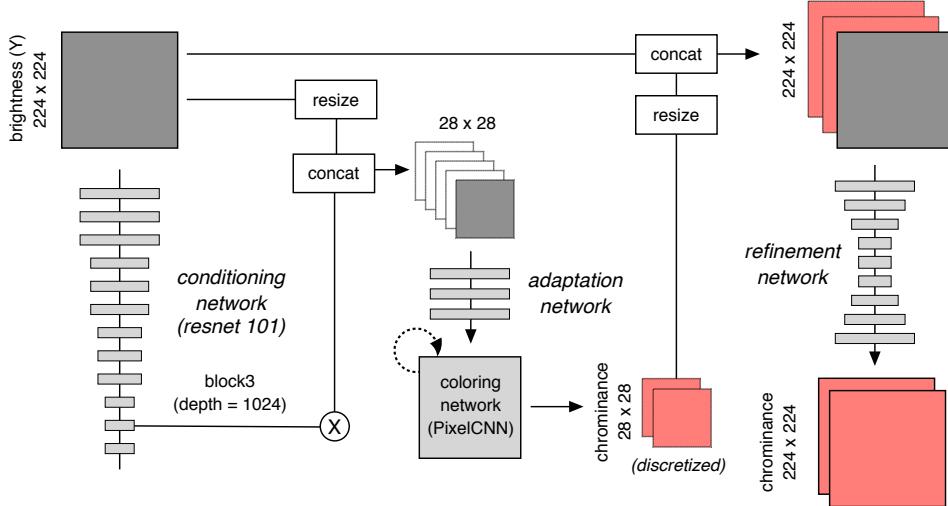


Figure 4: Diagram of Pixel Recursive Colorization (PixColor) method. We first pre-train the *conditioning network* on COCO image segmentation following [4]. Then, the conditioning network and the *adaptation network* convert the brightness channel  $Y$  into a set of features providing the necessary conditioning signal to the PixelCNN. The PixelCNN is optimized jointly with the conditioning and adaptation networks to predict a low spatial resolution version of the color image in a discretized space. The low spatial resolution image is subsequently supplied to a *refinement network*, which is trained to produce a full resolution colorization.

### 3 PIXEL RECURSIVE COLORIZATION (PIXCOLOR)

The key intuition behind our approach is that it suffices to predict a plausible *low resolution* color image, since color is much lower spatial frequency than intensity. To illustrate this point, suppose we take the *ground truth* chrominance of an image, downsample it to  $28 \times 28$ , upsample it back to the original size, and then combine it with the original luminance. Figure 3 shows some examples of this process. It is clear that the resulting colorized images look very close to the original color images.

In the sections below, we describe how we train a model to predict multiple plausible low resolution color images, and then how we train a second model to combine these predictions with the original grayscale input to produce a high resolution color output. See Figure 4 for an overview the approach.

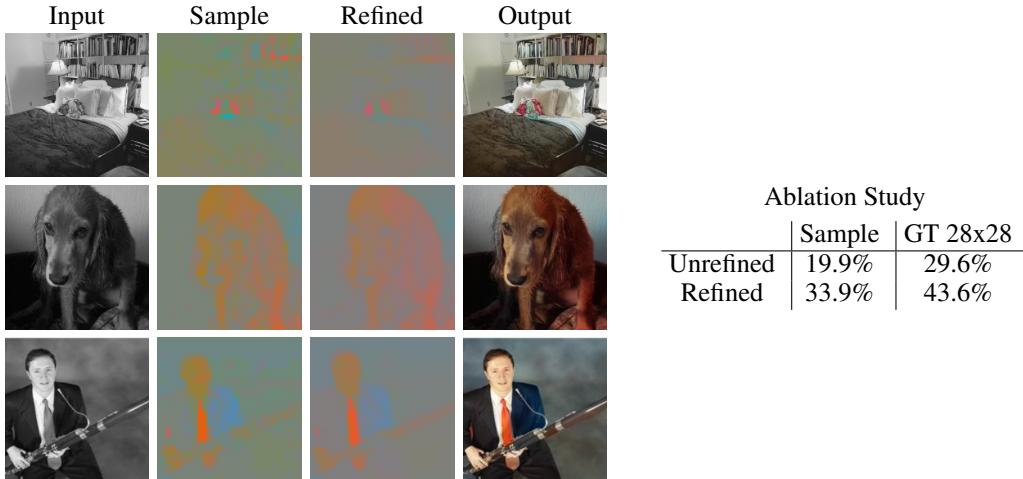


Figure 5: Left: The intermediate stages of PixColor. The column labeled "sample" is an output from PixelCNN, upsampled to the size of the image for visualization purposes. The column labeled "refined" is the output of the refinement network, before being combined with the grayscale input. Right: Results of a human evaluation using the "Visual Turing Test" metric explained in Section 4. We compare four systems: ground truth (GT) chroma image vs generated sample, passed directly into bilinear upsampling (unrefined) or passed into the refinement network.

### 3.1 PIXELCNN FOR LOW-RESOLUTION COLORIZATION

Inspired by the success of autoregressive models for unconditional image generation [35, 36] and super resolution [8], we use a conditional PixelCNN [36] to produce multiple low resolution color images. That is, we turn colorization into a sequential decision making task, where pixels are colored sequentially, and the color of each pixel is conditioned on the input image and previously colored pixels.

Although sampling from a PixelCNN is in general quite slow (since it is inherently sequential), we only need to generate a low-resolution image (28x28), which is reasonably fast. In addition, there are various additional speedup tricks we can use (see e.g., [27, 19]) if necessary.

Our architecture is based on [8] who used PixelCNNs to perform super resolution (another one-to-many problem). We use the YCbCr colorspace, because it is linear, simple and widely used (e.g., by JPEG). We discretize the Cb and Cr channels separately into 32 bins. Thus the model has the following form:

$$p(y|x) = \prod_i p(y(i,r) | y(1:i-1,:), x) p(y(i,b) | y(i,r), y(1:i-1,:), x)$$

where  $y(i,r)$  is the Cr value for pixel  $i$ , and  $y(i,b)$  is the Cb value. We performed some preliminary experiments using Logistic mixture models to represent the output values as suggested by the PixelCNN++ of [31], as opposed to using multinomials over discrete values [36]. However, we did not see a meaningful improvement, so for simplicity, we stick to a multinomial prediction model.

We train this model using maximum likelihood, with a cross-entropy loss per pixel. Because of the sequential nature of the model, each prediction is conditioned on previous pixels. During training, we "clamp" all the previous pixels to the ground truth values (an approach known as "teacher forcing" [39]), and just train the network to predict a single pixel at a time. This can be done efficiently in parallel across pixels.

### 3.2 FEEDFORWARD CNN FOR HIGH-RESOLUTION REFINEMENT

A simple way to use the low resolution output of the colorization network is to upsample it (e.g., using bilinear or nearest neighbor interpolation), and then to concatenate the result with the original

luminance channel. This can work quite well given groundtruth color, as we showed in Figure 3. However, it is possible to do better by learning how to combine the predicted low resolution color image with the original high resolution grayscale image.

For this, we use an image-to-image CNN which we call the refinement network. It is similar in architecture to the network used in [15] but with more layers in the decoding part. In addition, we use bilinear interpolation for upsampling instead of learned upsampling.

The refinement network is trained on a 28x28 downsampling of the ground truth chroma images. The reason we do not train it end-to-end with the PixelCNN is the following: the PixelCNN can generate multiple samples, all of which might be quite far from the true chroma image; if we forced the refinement network to map these to the true RGB image, it might learn to ignore these "irrelevant" color "hints", and just use the input grayscale image. By contrast, when we train using the true low-resolution chroma images, we force the refinement network to focus its efforts on learning how to combine these "hints" with the edge boundaries which are encoded in the grayscale image.

We show some qualitative examples of the benefits of the refinement network on the left of Figure 5. At first glance, the benefits seem small, but if you zoom in you will notice that the refinement network's outputs are much more plausible, since they better adhere to segment boundaries, etc. The results of a quantitative human evaluation of the refinement network, using the "Visual Turing Test" metric explained in Section 4, are shown in the table on the right of Figure 5. The increase from the Sample-Unrefined score (19.9%) to the Sample-Refined score (33.9%) shows the value added by the refinement network. The GT-Refined score (43.6%) shows the upper limit of our method could achieve with our refinement network (the maximum expected score for VTT is 50%).

## 4 EVALUATION METHODOLOGY

Since the mapping from gray to color is one-to-many, we cannot evaluate performance by comparing the predicted color image to the "ground truth" color image in terms of mean squared error or even other perceptual similarity metrics such as SSIM [37]. Instead, we follow the approach of [41] and conduct a "Visual Turing Test" (VTT) using a crowd sourced human raters. In this test, we present two different color versions of an image, one the ground truth and one corresponding to the predicted colors generated by some method. We then ask the rater to pick the image which has the "true colors". A method that always produces the ground truth colorization would score 50% by this metric.

To be comparable with [41], we show the two images sequentially for 1 second each. (We randomize which image is shown first.) Following standard practice, we train on the 1.2M training images from the ILSVRC-CLS dataset [30], and use 500 images from the "ctest10k" split of the 50k ILSVRC-CLS validation dataset proposed in [20]. Each image is shown to 5 different raters. We then compute the fraction of times the generated image is preferred to ground truth; we will call this the "VTT score" for short.

## 5 RESULTS

We assess the effectiveness of our technique by comparing against several recent colorization methods, both qualitatively and quantitatively. Table 3 shows a qualitative comparison of various recent methods applied to a few randomly chosen test images. Based on these examples, it seems that the best methods include our method (PixColor), and several recent CNN-based methods, namely LTBC [15], LRAC [20], and CIC [41]. Therefore, we conduct a more costly "Visual Turing Test" (VTT) on these four systems, as explained in Section 4.

Figure 6 summarizes the VTT scores. We see that our method significantly outperforms the previous state of the art methods, with an average VTT score of 33.9%.

One reason we think our results are better is that the colors they produce are more "natural", and are placed in the "right" places. To assess the first issue, Figure 7 plots the marginal statistics of the  $a$  and  $b$  channels (of CIELab) derived from the images generated from each image. We see that our model matches the empirical distribution (derived from the true color images) more closely than the

Method	LTBC	CIC	LRAC	PixColor (Seed 1)	PixColor (Seed 2)	PixColor (Seed 3)	PixColor (Oracle)
VTT (%)	25.8 ±0.97	29.2 ±0.98	30.9 ±1.02	33.3 ±1.04	35.4 ±1.01	33.2 ±1.03	<b>38.3</b> ±0.98

Figure 6: Results of the Visual Turing Test (VTT) study on the ImageNet test set. We report the fraction of times raters picked the generated color image over the ground truth with error ranges produced by bootstrapping the mean. Our study includes 500 test images and 5 raters per image. The column labeled "Oracle" is the score of the single best sample per image chosen by human raters.

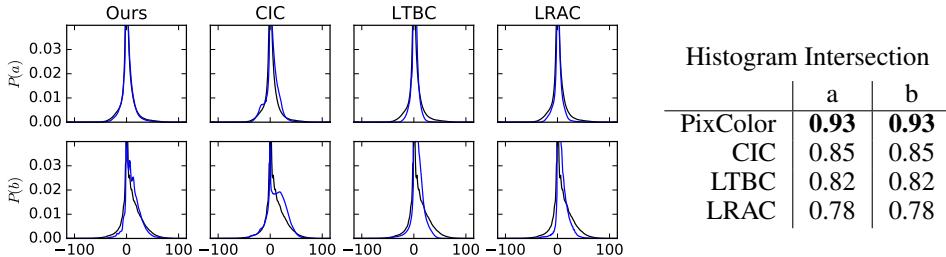


Figure 7: Marginal statistics of the color channels in Lab color space. Left: each method’s histogram is shown in blue against ImageNet’s test set histogram in black. Right: Histogram intersection on the color channels.

other methods, without needing to do any explicit reweighting of color bins, as was done in previous work [41].

### 5.1 SAMPLE DIVERSITY

Our model can produce multiple samples for each input, so for we run it 3 times, with 3 different seeds, and evaluate the outputs of each run independently. From Figure 6, we see that all of the samples are fairly good, but are they different from each other? That is, are the samples diverse?

Figure 2 suggests that our method can generate diverse samples. To quantitatively assess how different these samples are from each other, we compute the multiscale SSIM [37] measure between pairs of samples. The results are shown in Figure 8. We see that most pairs have an SSIM score in the 0.95-0.99 range, meaning that they are very similar, but differ in a few places, corresponding to subtle details, such as the color of a person’s shirt. The pairs which have the lowest SSIM score are the ones where large objects are given different colors (see the pair of birds on the left hand side).

In an ideal world, we could automatically select the single best sample, and just show that to the user. To get a sense of how well this could perform, we decided to use humans to perform the task of picking the best sample. More precisely, for each of the 3 samples for a given image, we picked the one that the most raters liked. We then computed the VTT score for these single samples using a different set of raters. The VTT score jumps to 38.3%. This suggests that an algorithmic way to pick a good sample from the set could yield significantly better results.

We did some preliminary experiments where we used the likelihood score (according to the Pixel-CNN model) to pick the best sample, but this did not yield good correlation with human judgement. It may be possible to train a separate ranking model, but we leave that to future work.

## 6 CONCLUSION

We showed PixColor produces diverse colorizations and found that on average the outputs of our model perform better than other published methods in a crowd sourced human evaluation. We avoid the problem of slow inference in PixelCNN by only sampling low-resolution color channels and use a standard image-to-image CNN to refine the result. We justified the necessity of the refinement network with ablation studies and we showed that PixColor outputs more closely match the marginal

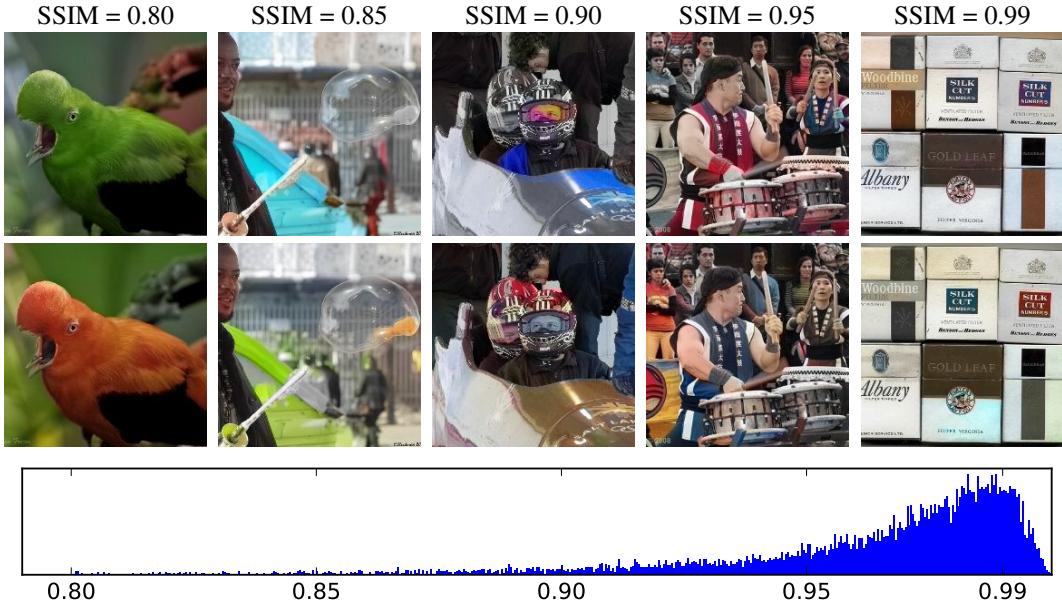


Figure 8: To demonstrate that our model produces diverse samples, we compare two outputs from the same input with multiscale SSIM. A histogram of the SSIM distances from the ImageNet test set is shown above. Representative pairs are shown at various SSIM distances. An SSIM value of 1.0 means the images are identical.

color distributions when compared to other methods. The model exhibits a variety of failure modes, as illustrated in Figure 10, which we will address in our future work.

#### ACKNOWLEDGMENTS

We thank Stephen Mussmann and Laurent Dinh for work and discussion on earlier versions of this project; Julia Winn, Jingyu Cui and Dhyanesh Narayanan for help with an earlier prototype; Aäron van den Oord for advice and guidance employing PixelCNN architectures; the TensorFlow team for technical and infrastructure assistance.

#### REFERENCES

- [1] Mohammad Haris Baig and Lorenzo Torresani. Multiple hypothesis colorization and its application to image compression. *Comput. Vis. Image Underst.*, 2017.
- [2] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. *arXiv:1702.06674*, 2017.
- [3] G. Charpiat, M. Hofmann, and B. Schölkopf. Automatic image colorization via multimodal predictions. *ECCV*, 2008.
- [4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.
- [5] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. *arXiv:1605.00075*, 2016.
- [6] Alex Yong-Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu-Wing Tai, Siu-Yeung Cho, Ping Tan, and Stephen Lin. Semantic colorization with internet images. *ACM Transactions on Graphics*, 30(6), 2011.
- [7] Ryan Dahl. Automatic colorization. <http://tinyclouds.org/colorize/>, 2016.

- [8] Ryan Dahl, Mohammad Norouzi, and Jonathon Shlens. Pixel recursive super resolution. *arXiv:1702.00783*, 2017.
- [9] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, and David A. Forsyth. Learning diverse image colorization. *arXiv:1612.01958*, 2016.
- [10] Aditya Deshpande, Jason Rock, and David Forsyth. Learning large-scale automatic image colorization. *ICCV*, 2015.
- [11] Kevin Frans. Outline colorization through tandem adversarial networks. 28 April 2017.
- [12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 2014.
- [13] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. *ACM Multimedia*, 2012.
- [14] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. *ACM Multimedia*, 2005.
- [15] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics*, 35(4), 2016.
- [16] Revital Irony, Daniel Cohen-Or, and Dani Lischinski. Colorization by example. *Eurographics Symposium on Rendering*, 2005.
- [17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv:1611.07004*, 2016.
- [18] Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. *ICLR*, 2014.
- [19] Alexander Kolesnikov and Christoph H Lampert. Latent variable PixelCNNs for natural image modeling. *arXiv:1612.08185*, 2016.
- [20] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. *ECCV*, 2016.
- [21] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3), 2004.
- [22] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. *Eurographics Symposium on Rendering*, 2007.
- [23] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv:1611.02163*, 2016.
- [24] Yuji Morimoto, Yuichi Taguchi, and Takeshi Naemura. Automatic colorization of grayscale images using multiple images on the web. *SIGGRAPH*, 2009.
- [25] François Fleuret, Anil C. Kokaram, and Rozenn Dahyot. Automated colour grading using colour distribution transfer. *Comput. Vis. Image Underst.*, 107(1), 2007.
- [26] Y Qu, T Wong, and P Heng. Manga colorization. *ACM Transactions on Graphics*, 25(3), 2006.
- [27] Prajit Ramachandran, Tom Le Paine, Pooya Khorrami, Mohammad Babaeizadeh, Shiyu Chang, Yang Zhang, Mark A Hasegawa-Johnson, Roy H Campbell, and Thomas S Huang. Fast generation for convolutional autoregressive models. *ICLR workshop*, 2017.
- [28] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, 21(5), 2001.
- [29] Amelie Royer, Alexander Kolesnikov, and Christoph H Lampert. Probabilistic image colorization. 11 May 2017.

- [30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.
- [31] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. *ICLR*, 2017.
- [32] D Sykora, J Burianek, and J Zara. Unsupervised colorization of black-and-white cartoons. *International symposium on Non-photorealistic animation and rendering*, 2004.
- [33] Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. *CVPR*, 2005.
- [34] S Tsaftaris, F Casadio, J Andral, and K Katsaggelos. A novel visualization tool for art history and conservation: Automated colorization of black and white archival photographs of works of art. *Studies in Conservation*, 59(3), 2014.
- [35] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv:1601.06759*, 2016.
- [36] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *arXiv:1606.05328*, 2016.
- [37] Z Wang, E P Simoncelli, and A C Bovik. Multiscale structural similarity for image quality assessment. *Asilomar Conference on Signals, Systems, Computers*, 2003.
- [38] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. *ACM Transactions on Graphics*, 21(3), 2002.
- [39] R J Williams and D Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2), 1989.
- [40] L. Yatziv and G. Sapiro. Fast image and video colorization using chrominance blending. *IEEE Trans. Img. Proc.*, 15(5), 2006.
- [41] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. *ECCV*, 2016.
- [42] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned deep priors. *SIGGRAPH*, 2017.

## APPENDIX

	Operation	Kernel	Strides	Feature maps
<b>PixelCNN conditioning network</b>	224 × 224 × 3 input			
Conv2D	7 × 7	2		64
3 × ResNet block	3 × 3	2, 1, 1		64/256 bottleneck
4 × ResNet block	3 × 3	2, 1, 1, 1		128/512 bottleneck
23 × ResNet block	3 × 3	1		256/1024 bottleneck
3 × Conv2D	3 × 3	1		64
Gradient Multiplier	Engaged at 100,000 steps			
<b>PixelCNN colorization network</b>	28 × 28 × 2 input			
Masked Conv2D	7 × 7	1		64
10 × Gated Conv2D Blocks	5 × 5	1		64
Masked Conv2D	1 × 1	1		1024
Masked Conv2D	1 × 1	1		2 * 32
<b>Refinement Network</b>	224 × 224 × 3 input			
Conv2D	3 × 3	2		64
2 × Conv2D	3 × 3	1, 2		128
2 × Conv2D	3 × 3	1, 2		256
2 × Conv2D	3 × 3	1		512
Conv2D	3 × 3	1		256
4 × Conv2D	3 × 3	2, 1, 2, 1		512
Conv2D	5 × 5	1		1024
Conv2D	3 × 3	1		512
2 × Conv2D	3 × 3	1		128
Bilinear Upsample				
2 × Conv2D	3 × 3	1		64
Bilinear Upsample				
2 × Conv2D	3 × 3	1		32
Conv2D	1 × 1	1		2
Optimizer	Adam (beta1=0.9, momentum=0.9)			
Batch size	64 = 8 × 8 (8 GPUs, synchronous updates)			
Iterations	360,000			
Learning Rate	0.0003			
Colorness Threshold	0.05			
Weight, bias initialization	Truncated normal (stddev=0.1), Constant(0)			

Table 2: Hyperparameters used. The PixelCNN and Refinement networks were trained independently.

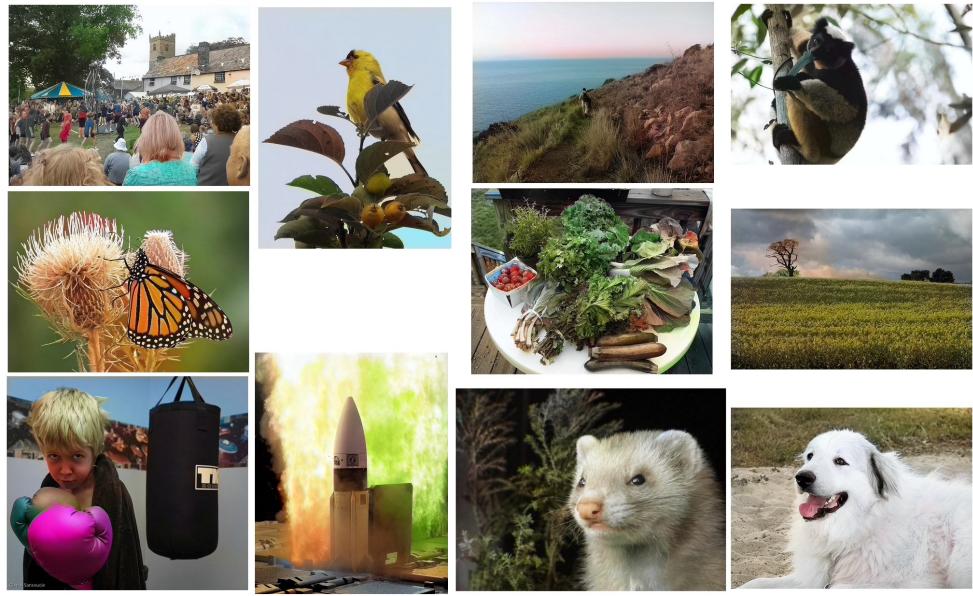


Figure 9: Selected high resolution and non-square samples.

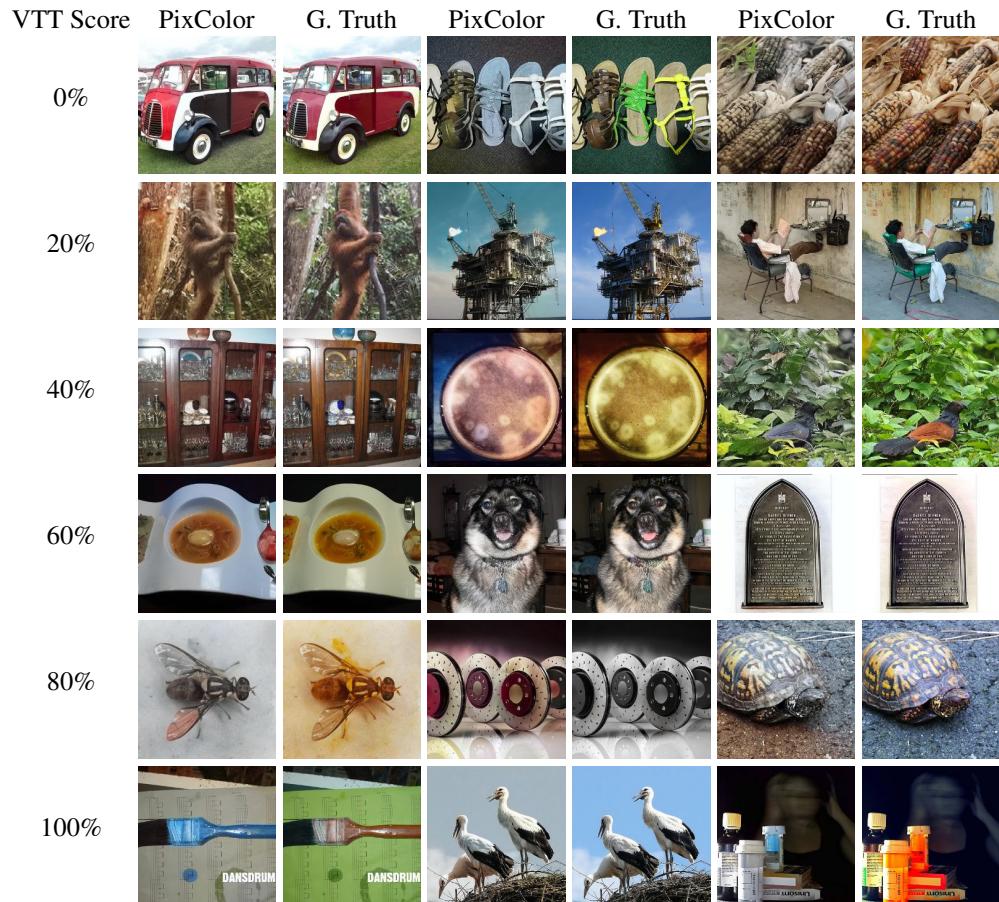


Figure 10: Images illustrating each possible VTT score.

Table 3: Qualitative side by side comparison of colorizations produced by various methods (LTBC: [15], pix2pix: [17], VAE: [9], LRAC: [20], CIC: [41], PixColor: this paper, G. Truth: original color). These images are *randomly sampled* from the ImageNet test set.

