DEEP AUTORESOLUTION NETWORKS

Gabriel Pereyra & Christian Szegedy
Google
{pereyra,szegedy}@google.com

ABSTRACT

Despite the success of very deep convolutional neural networks, they currently operate at very low resolutions relative to modern cameras. Visual attention mechanisms address this by allowing models to access higher resolutions only when necessary. However, in certain cases, this higher resolution isn't available. We show that autoresolution networks, which learn correspondences between low-resolution and high-resolution images, learn representations that improve low-resolution classification - without needing labeled high-resolution images.

1 INTRODUCTION

Despite the success of very deep convolutional neural networks in machine vision (Simonyan & Zisserman (2014); Szegedy et al. (2014); He et al. (2015)), the resolution at which they operate is still less than 1% that of an iPhone camera [1]. Applying vanilla convolutional neural networks directly to high-resolution images is currently computationally infeasible but can be addressed using visual attention models (Ranzato (2014)). However, these models currently require access to high-resolution images to ”zoom” into.

To address the case when access to labeled high-resolution images is limited, we propose autoresolution networks. Similar to Doersch et al. (2015), we define an unsupervised context prediction task - predict whether a high-resolution patch belongs to a low-resolution image or where the high-resolution patch corresponds to the low-resolution image. This formulation allows us to make hundreds of predictions per image, resulting in the ability to train very deep unsupervised convolutional neural networks. We show that we can train a model with more than forty convolutional layers and that the representations learned improve an already strong low-resolution classification (Krizhevsky (2009)) baseline.

2 MODEL

Our architecture is motivated by the success of the inception architecture (Szegedy et al. (2014)) and siamese networks (Chopra et al. (2005)). Our model consists of two convolutional towers, a patch embedding tower which consists of 17 convolutional layers and a low-resolution image embedding tower which consists of 40 convolutional layers, that are fed a high-resolution patch and a low-resolution image. We concatenate the outputs of both convolutional towers, and feed this to a classifier or regressor, that consists of 5 convolutional layers. The convolutional tower that processes images uses primarily inception layers while the convolutional tower that processes patches uses primarily convolutional layers. All layers use rectified linear activations, except for the final convolutional layer, which has a softmax activation in the case of classification or no activation in the case of regression. To our knowledge, this is the deepest unsupervised model trained to date.

For the patch classification objective, the model must predict whether the high-resolution patch belongs to the low-resolution image. In our experiments, for every batch, our model predicted a positive example, where the high-resolution patch was sampled from the image, and a negative example, where the high-resolution patch was sampled from another image. To create the negative example, we incremented the patch's index within the batch by one so that they were no longer aligned with the image they were sampled from. For the patch regression objective, The model must

[1]Assuming commonly used 256x256 input and iPhone 6's 8-megapixel camera.

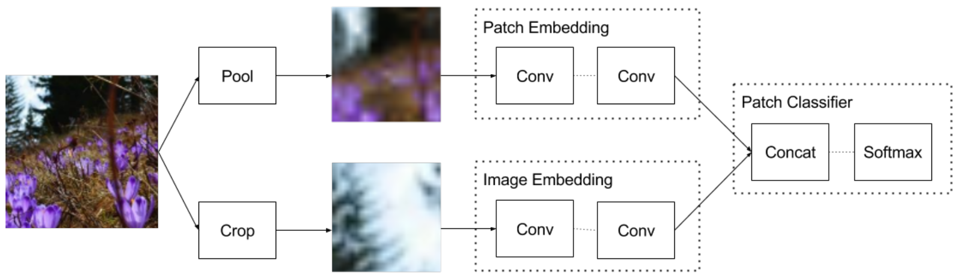

Figure 1: Starting from a high-resolution image (far left), our model creates a low-resolution version through average pooling and a high-resolution patch by taking a random crop of the original image. Each of these two new images is embedding by a series of convolutions and the embeddings are concatenated and fed to a patch classifier. The classifier must predict whether the patch belongs to the image (negatives are created by sampling patches from other images in the batch). We also trained a regression variant which must instead predict the coordinates of where the patch was cropped (not shown).

predict the coordinates of where the high-resolution image was cropped from the low-resolution image. In our experiments, we used the center coordinates of a patch normalized to ±1. For both objectives we didn't find our model relied on trivial solutions or chromatic aberrations (Doersch et al. (2015)).

3 Experiments

To evaluate our model, we first optimized both unsupervised objectives using an internal Google dataset consisting of 100M images similar to ImageNet. Then, we used the convolutional tower that processes the low-resolution images to initialize a classifier on the Cifar10 dataset.

3.1 Hyperparameters

All models were implemented in TensorFlow (Abadi et al.) and trained using NVidia Kepler GPUs. For optimization, we used RMSprop (Tieleman & Hinton (2012)) with decay of 0.9, $\epsilon = 1.0$, and a batch size of 32. All learning rates were decayed by 0.94 every 2 epochs and gradients were clipped (Pascanu et al. (2012)) to 10 based on their global norm. All models were evaluated using a running average of their parameters.

For the patch classification objective, our model used a learning rate of 0.003. For the patch regression objective, our model used a learning rate of 0.01. For Cifar10, we trained a baseline using a learning rate of 0.01. We fine-tuned models initialized with the weights of converged patch classification and regression models using learning rates [0.01, 0.003, 0.001, 0.0003], and found 0.003 to work best for the patch classification model and 0.001 for the patch regression model.

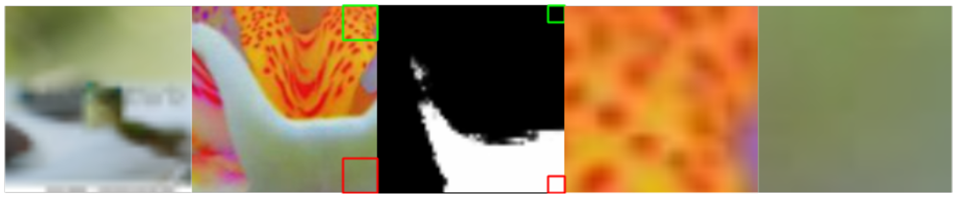

Figure 2: Given a low-resolution image (far-left) and a high-resolution "super" patch from another image (right), our model produces a binary heat map (middle). The green and red boxes show which part of the high-resolution patch correspond to a binary value in the heat map. The right and far-right image show what the receptive field of the patch embedding sees when it makes a correct (green) and incorrect (red) classification.

Table 1: Cifar10 error for a randomly initialized model, and models initialized with the converged weights of the image tower trained to optimize the patch classification and regression objective.

Models	**Error**
No Fine-Tuning	9.09%
Fine-Tuned (Regression)	6.94%
Fine-Tuned (Classification)	**6.60%**

3.2 Tiled Predictions

For our unsupervised experiments, we found that making multiple predictions per image substantially improved the convergence of our models. Concretely, we crop a $127 \times 127 \times 3$ "super patch" randomly from a $256 \times 256 \times 3$ image. By unrolling the patch convolutional tower over the "super" patch we can make hundreds of predictions per image over a grid of overlapping, adjacent patches, instead of a single prediction. The receptive field of the convolutional tower that processed the patches meant that for every image, we made 47×47 binary predictions for the patch classification objective or $47 \times 47 \times 2$ real valued predictions for the patch regression objective.

3.3 Cifar10

Using the low-resolution embedding tower, which consisted of 40 convolutional layers, we add a softmax layer with 50% dropout to create a classifier. To obtain a baseline, we trained this model with random initializations. Then, we initialized the same model using the weights of converged patch classification and patch regression models. We found this form of pre-training (Bengio et al. (2007)) improved an already strong Cifar10 (Krizhevsky (2009)) baseline. For all models, we trained on 45K training images, continuously validated on the remaining 5K images in the training set, and evaluated the best model only once on the test set. For the fine-tuned models, we performed 10K steps of gradient descent with the initializations fixed, to update only the softmax layer, to prevent its random initialization from destroying the learned initialization.

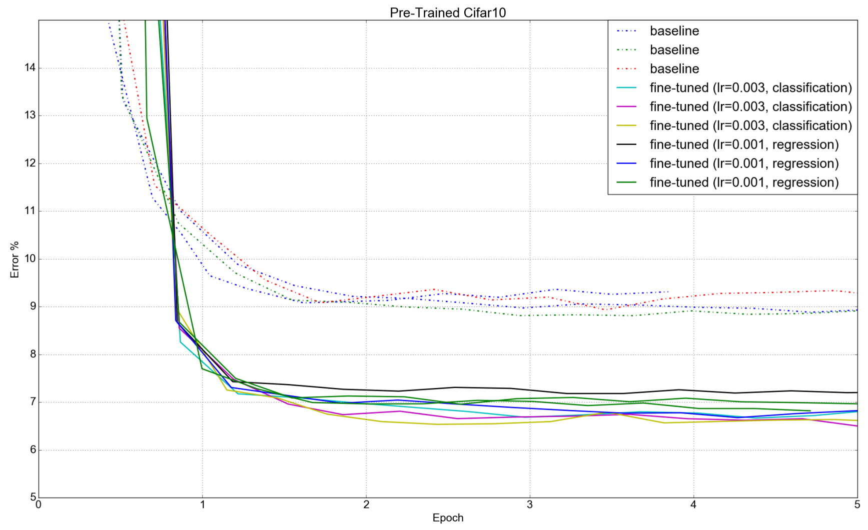

Figure 3: Cifar10 validation error of randomly initialized and models pre-trained with autoresolution networks.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, Universit De Montral, and Montral Qubec. Greedy layer-wise training of deep networks. In *In NIPS*. MIT Press, 2007.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pp. 539–546, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.202. URL `http://dx.doi.org/10.1109/CVPR.2005.202`.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015. URL `http://arxiv.org/abs/1505.05192`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

Marc'Aurelio Ranzato. On learning where to look. *CoRR*, abs/1405.5488, 2014. URL `http://arxiv.org/abs/1405.5488`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop. *COURSERA: Neural networks for machine learning*, 2012.