Image Inpainting using Auto-encoder and CNN

Ishay Tratner and Itamar Almog

Abstract:

In this paper, we propose a learning-based approach for automatic filling of holes in pictures using autoencoders and convolutional neural networks. Autoencoder is a method that compresses data and then decompresses it, on the decompression it makes new pixels, the cnn is used in the network to compress the data and also decompress it, cnn can retain the connected information between the pixels of an image.

With that technique of autoencoder we can build a model that will be aware of context details that are called Context Encoders that are significantly more effective in capturing high-level features than prior techniques.

Introduction:

Before someone shares a photo he maybe wants to erase the object, or recover the image content in occluded image areas. These require automated hole filling, a challenge that steel active because its ambiguity and the complexity of natural images, general hole-filling remains challenging.

There are three main methods :

1. Fills the hole by extending textures from surrounding regions. (algorithm)
2. Predict missing image regions in a data-driven fashion, leveraging large external databases.
3. Create new data using the rest of the picture.

Fills the hole by extending textures from surrounding regions:

That method always looks for better matching patches. There are cons in that method because they do not capture the semantics or global structure of the image.

Predict missing image regions in a data-driven fashion, leveraging large external databases:

In that method assume that regions surrounded by similar context likely possess similar content. That method could fail when the query image is not well represented in the database. That method cons is that method requiring access to the external database, which greatly restricts possible application scenarios.

Create new data using the rest of the picture:

Compute a vector that represents the rest of the picture, and from that vector create the missing data.

Related work:

Chao Yang Xin Lu Zhe Lin Eli Shechtman Oliver Wang and Hao Lik, High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis 2017

Project description:

In our project like Implied before we used a trained encoder-decoder CNN (Context Encoder). We started to work on that method before we knew about their pros or cons. We just knew that it is possible to inpainting pictures like that and we wanted to try.

First we took the pictures and separate them to 90% train and 10% test, after that we take the train pictures and we did hole in the middle of the picture before we copied the full picture because we need to train the model on complete picture and picture with hole because he just need to learn how to complete the image.
After that we take the test picture and make holes in the middle like before and we put them to our trained encoder-decoder CNN now without the full picture because it is a test.

The continuation of this section is in the **our method** section.

Previous attempts:

Linear regression:

In that method we inpating just one “dead” pixel in math of 3x3, we used in linear regression because we needed to estimate what was the dead pixel and that method intended to estimate values and we needed a number between 0-255.

Recall that the problem with this method was such blurring and fireflies.

We tried to make logistic regression to 255 classes, but it failed (it was also very slow).

Neural network:

We discovered from the earlier method that the problem was the amount of data needed to reconstruct the dead pixel, so we took a bigger radius around the dead pixel and then we ran it again and we saw the effect of the hidden layers. The effect of the hidden layers was that the picture came out without any noise and sharper than earlier.

Comparison (fireflies and artifacts):

Linear regression: Neural network: Original:

all that methods was for inpainting one pixel and in the current method we try inpainting

Our method:

The input to a context encoder is an image with one of its regions “dropped out”, i.e., set to zero. The removed regions could be at any place and any size, we present two different strategies here:

Central region: the simplest such shape is the central square patch in the image Figure 1.

Different place and size: is more complicated mainly for cutting and reassembling the image.

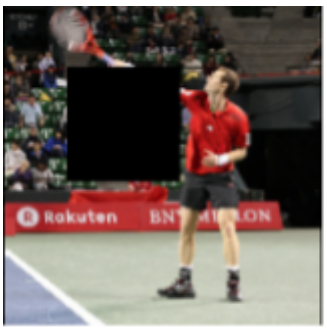


Figure 1

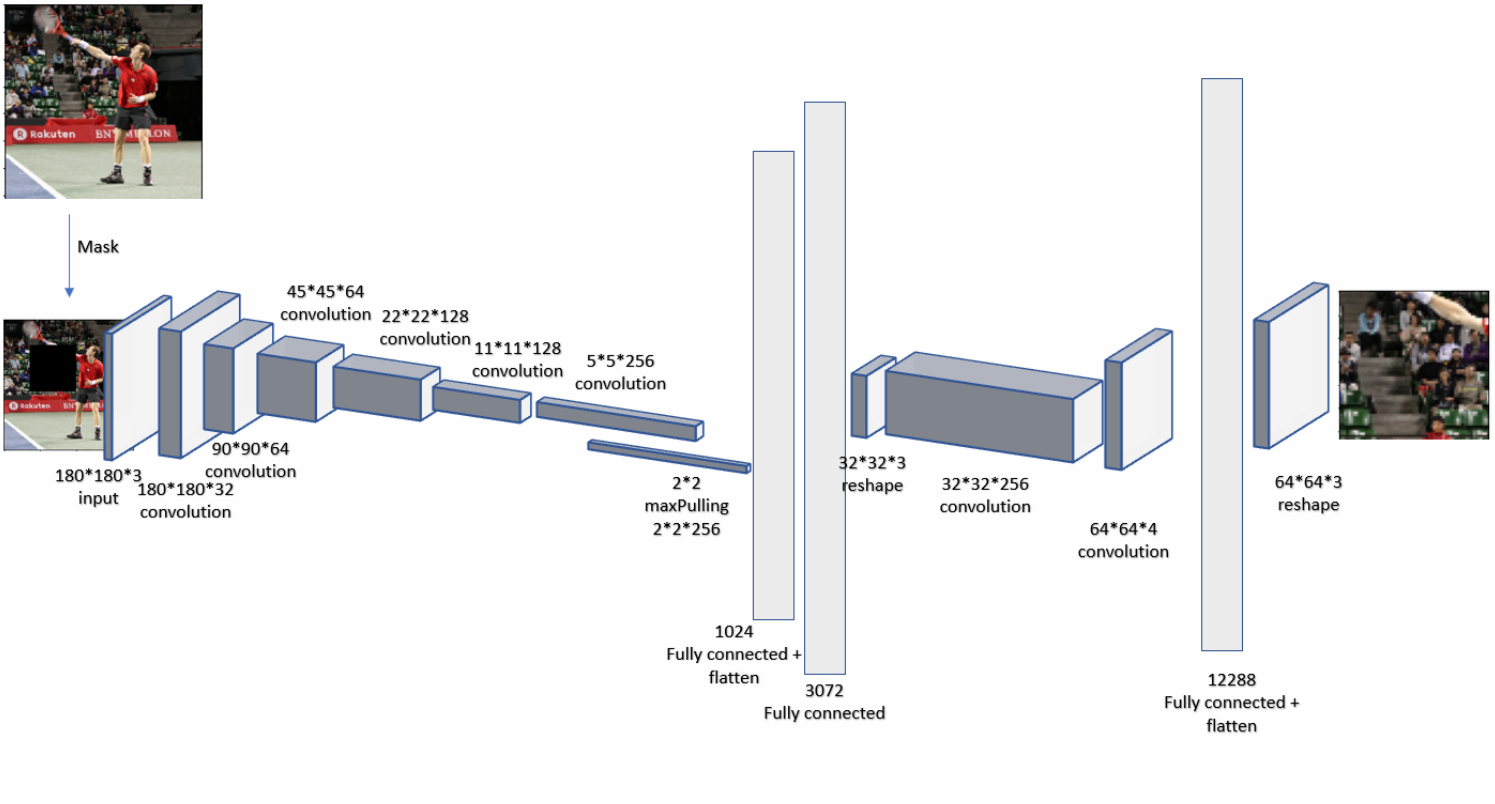


Figure 2

Encoder-decoder pipeline:

The overall architecture is a simple encoder-decoder pipeline. The encoder takes an input image with missing regions and produces a latent feature representation of that image. The decoder takes this feature representation and produces the missing image content. We found it important to connect the encoder and the decoder through a fully-connected layer, which allows each unit in the decoder to reason about the entire image content. Figure 2 shows an overview of our architecture.

Encoder:

Our network is trained for context prediction by the rest of the picture.

Given an input image of size 180x180, we use the first five convolutional layers and the following pooling layers to compute an abstract 2x2x256 dimensional feature representation. On every step of convolution and maxPulling it makes a picture with less resolution but with more dimensions in order to catch the shapes of the picture.

However, if the encoder architecture is limited only to convolutional layers, there is no way for information to directly propagate from one corner of the feature map to another. This is so because convolutional layers connect all the feature maps together, but never directly connect all locations within a specific feature map. In the

present architectures, this information propagation is handled by fully connected or inner product layers, where all the activations are directly connected to each other. In our architecture, the latent feature dimension is 2x2x256 = 1024, then we upscale that with fully connected to 3072.

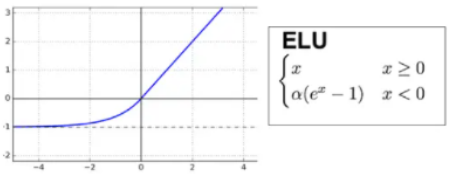
Because we don't need to make a picture from random noise like any autoencoder, we made the encoded matrix big, but because we have fully connected, if we made it bigger the parameter count exploded - more then 1,000M (1B), so we tried to keep that small.

Decoder:

We now discuss the second half of our pipeline, the decoder, which generates pixels of the image using the encoded features. The encoded features are connected to the decoder using a fully connected layer.

The fully-connected layer is followed by a series of two convolutional layers with learned filters, each with a Exponential Linear Unit (ELU) activation function. In between the convolutions there is an up sample layer that upsampling the picture to a bigger one. at the reconstruction of the end picture there is the second convolution that her size is 64*64*4, so it has more channel then needed for feature reconstruction, so we pass it to a fully connected network, that make it 64*64*3.

We used ELU activation because it can handle infinity minus numbers well, even better than leaky ReLu and because we want to keep the numbers closer to positive numbers the ELU function minimum if -1.

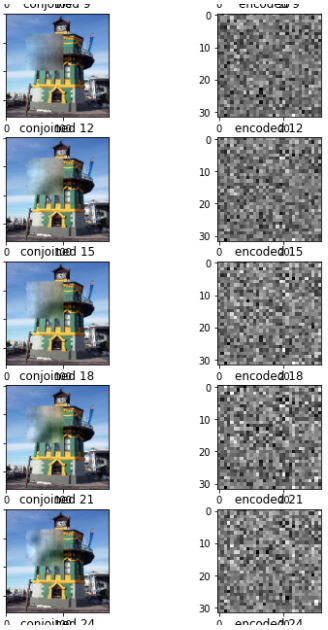


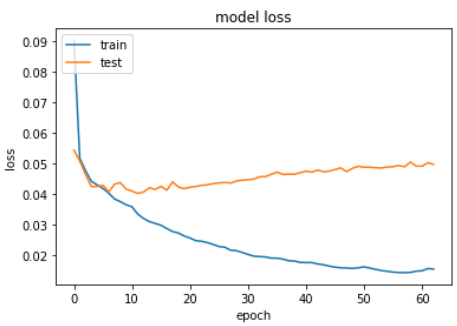
Training:

We trained the network on a 180x180 sized 4,500 pictures and on hole size of 64x64, that is about ⅛ of the picture, if we made the hole smaller the results are even better.

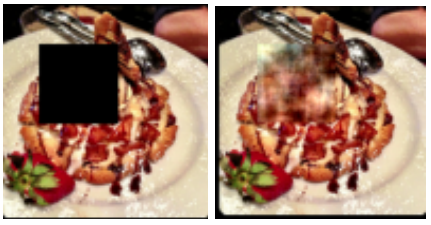
The picture on the right **during** the training shows the progress and how the encoded matrix looks like.

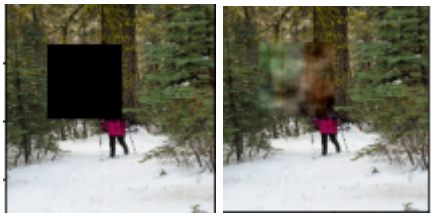
We found that the loss if the validation doesn't matter and that the more the training loss is smaller the more it makes the final picture sharper.





Some results:





Conclusion:

autoencoder are great at making photos that does not exist but it main drawback are:

1. need to retrain for every size of picture or hole or reposition of the hole.
2. The output is really blurry.

There are ways to make the autoencoder better like: training the network with similar pictures to what you want to complete, or even training on more pictures (bigger dataset).

Maybe GAN can do a better job than an autoencoder because its output is sharper and more robust.