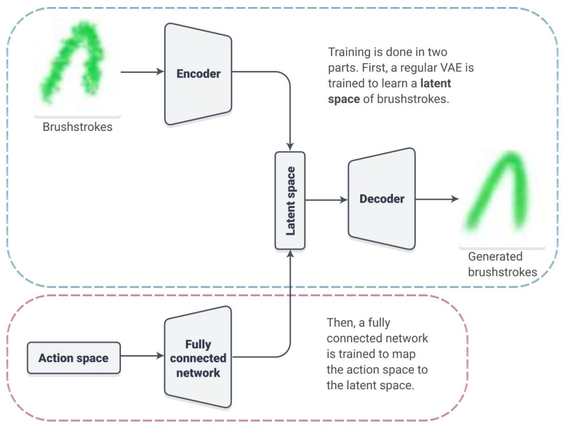
# Q: How do we get neural nets to paint?
# A: Give it a differentiable paintbrush!

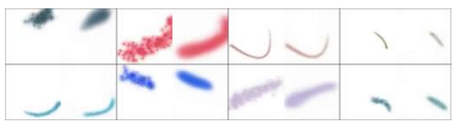
**Neural Painters: a learned differentiable constraint for generating brushstroke paintings** by Reiichiro Nakano

## Step 1: Create a paintbrush by Training a differentiable model of brushstrokes

There are numerous painting programs, but most are *non-differentiable*. Since it's difficult to work with non-differentiable things in the context of deep learning, we'll train a neural network (the **neural painter**) to *simulate* an external painting program (MyPaint). A neural painter is a **learned differentiable painting program**.
We explore 2 ways to train a neural painter.

### VAE Neural Painter

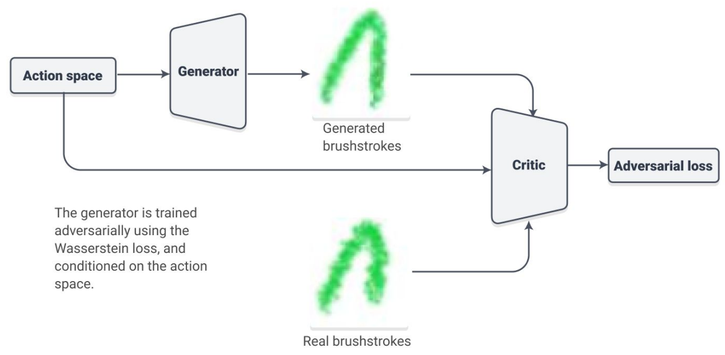


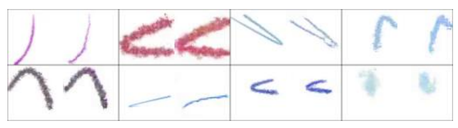



Left: Real  Right: Generated

A VAE neural painter results in brushstrokes that are slightly blurred and *smudged*.

### GAN Neural Painter



Left: Real  Right: Generated

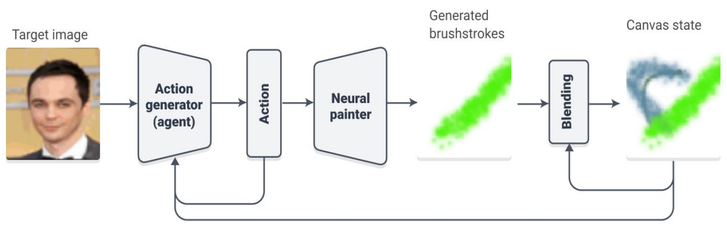
A GAN neural painter results in brushstrokes that emulate the original texture and look more realistic.

## Step 2: Use the neural painter to paint! (Multiple ways)

### 2a: Train a separate agent to reconstruct any target image with brushstrokes

One way to use the neural painter is to train a separate network (the agent) to reconstruct any given target image using brushstrokes. We use an adversarial training framework, with a (1) **generator/agent** conditioned to reconstruct an input image with brushstrokes from a neural painter, and a (2) **critic** deciding how well the generated painting matches the target. Training is fully differentiable and done with backpropagation, but at evaluation, the brushstrokes generated by the agent can be transferred back to the real non-differentiable painting program. In this way, the method can be seen as a form of model-based reinforcement learning with a learned world model.

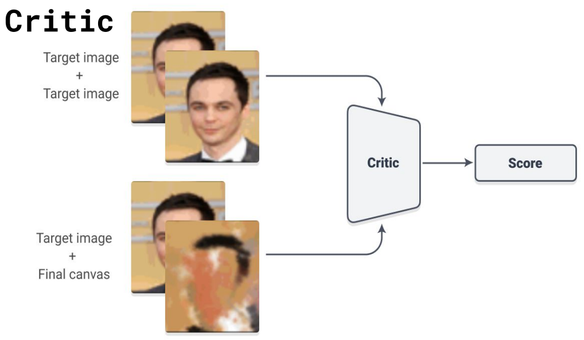
#### Generator




The recurrent agent generates a sequence of actions based on a target image, and successively applies them to a canvas.

Only the final canvas state after all strokes have been generated is used for adversarial training of the agent.

#### Critic




Conditioning on a target image is achieved by concatenating the target image with both itself and the final canvas state before feeding it into the critic.

#### Results

Left: Target image

Center: Generated painting with VAE neural painter-trained agent

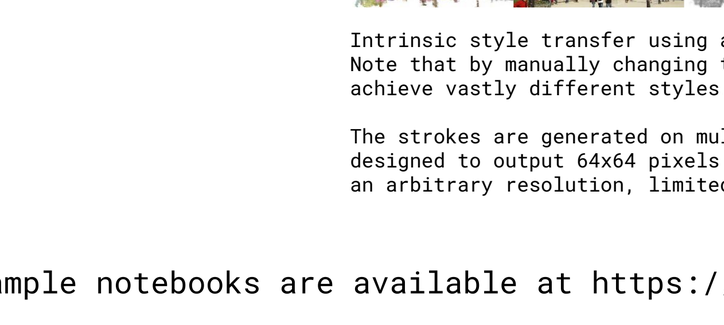
Right: Generated painting with GAN neural painter-trained agent

### 2b: Optimize brushstrokes to maximize ImageNet classes in a pre-trained image classifier

Since the neural painter is differentiable, we can directly use gradient descent to find a set of brushstrokes that minimize some loss function. For example, we can use gradient ascent to find a set of brushstrokes that *maximizes* the probability of a particular class in a pre-trained ImageNet classifier. A fun way to interpret the results of maximizing the probability of the *panda* class is as the answer to the question:

***Given a brush, how would a pre-trained ImageNet classifier paint the optimal panda?***

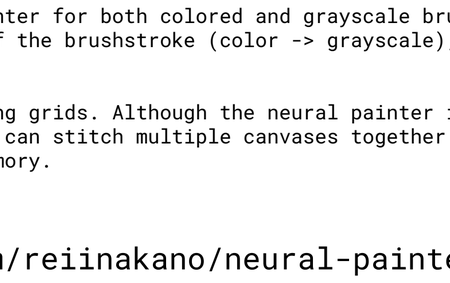
Optimizing multiple ImageNet classifiers at the same time (e.g. ResNet, GoogleNet, VGG) can result in better outputs.

Optimal pandas

Optimal bees

Optimal violins



Other ImageNet class visualizations

### 2c: Intrinsic style transfer: Optimize brushstrokes to minimize content loss in neural style transfer

In intrinsic style transfer, we use gradient descent to find a set of brushstrokes that minimize neural style transfer's content loss for a target content image. Intuitively, this method finds brushstrokes that preserve only the meaningful higher-level content in the target image. The *style* is an **intrinsic** property dictated purely by the artistic medium, in this case, brushstrokes.






Intrinsic style transfer using a GAN neural painter for both colored and grayscale brushstrokes. Note that by manually changing the primitives of the brushstroke (color -> grayscale), we can achieve vastly different styles.

The strokes are generated on multiple overlapping grids. Although the neural painter is only designed to output 64x64 pixels on a canvas, we can stitch multiple canvases together to achieve an arbitrary resolution, limited only by GPU memory.

## Step 3: Try it yourself! An easy-to-use library and sample notebooks are available at https://github.com/reiinakano/neural-painters-pytorch