

Each neuron in a neural network gets activated by specific patterns in the input image, defined by the weights in its receptive field.

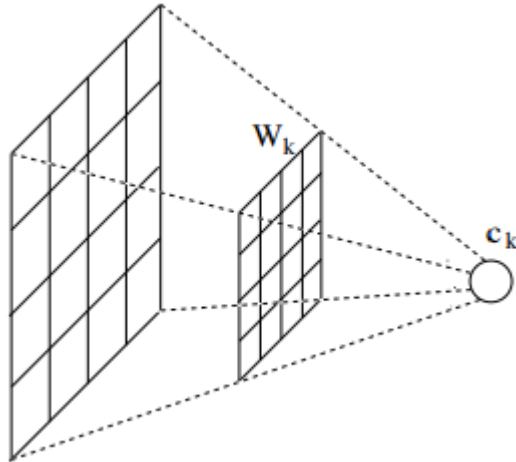
The intuition is that neurons at higher layers should recognize increasingly complex patterns, obtained as a combination of previous patterns, over a larger receptive field.

In the highest layers, neurons may start *recognizing patterns* similar to *features of objects* in the dataset, such as feathers, eyes, etc. In the final layers, neurons get activated by “patterns” identifying objects in the category.

Can we confirm such a claim?

Visualization of hidden layers

Our goal: find a way to *visualize the kind of patterns a specific neuron gets acti-*




vated by.

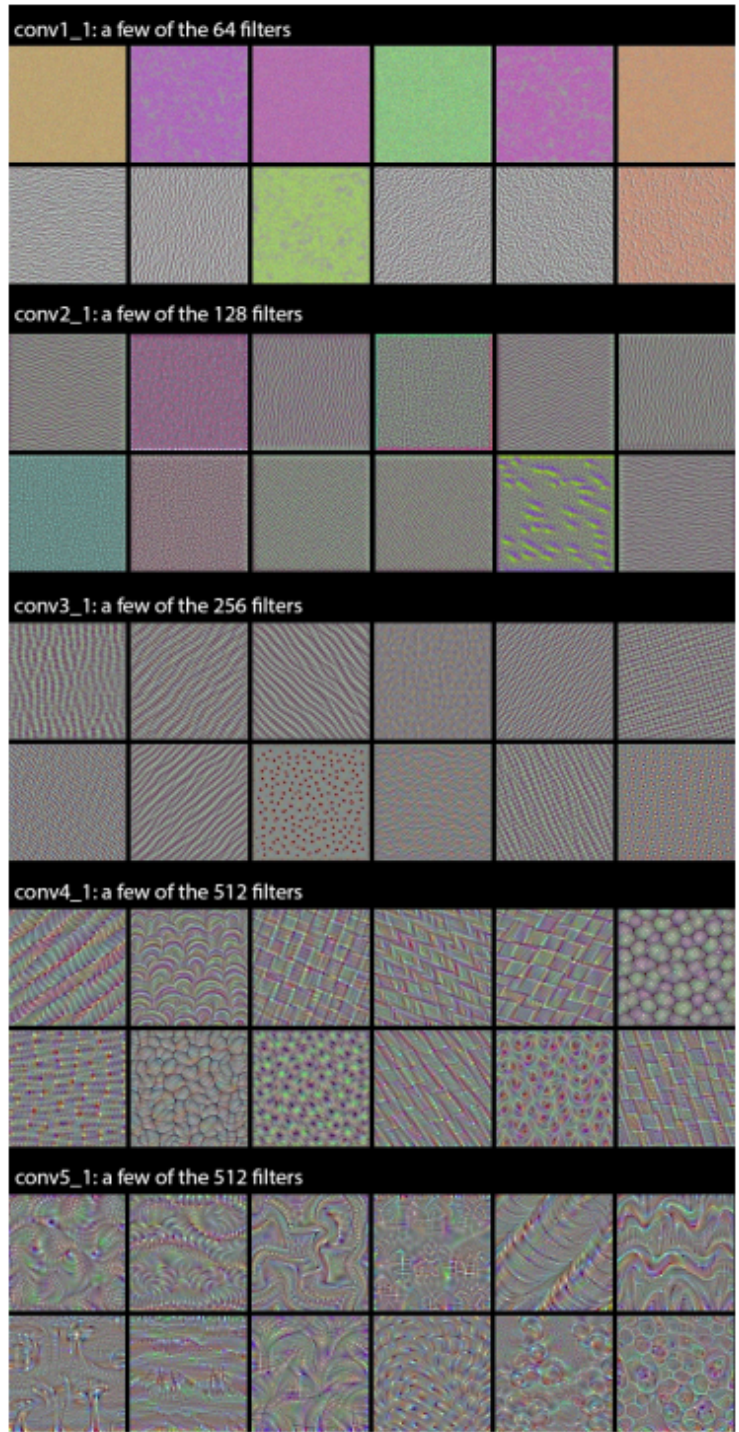
The loss function $L(\theta, x)$ of a NN depends on the *parameters* θ and the *input* x . During training, we fix x and compute the partial derivative of $L(\theta, x)$ w.r.t the parameters θ to adjust them in order to decrease the loss. In the same way, *we can fix* θ and use partial derivatives w.r.t. input pixels in order to synthesize images *minimizing the loss*. In this way, we can compute an activation of some neuron, and understand how I should modify my input to increase or decrease the activation of a neuron. Thus, we could find, for example, which kind of `_input` maximizes this kind of activation.

The gradient ascent technique



Start with a random image, e.g.  - do a *forward pass* using this image x as input to the network to *compute the activation* $a_i(x)$ caused by x at some neuron (or at a whole layer) - do a *backward pass* to compute the gradient of $\frac{\partial a_i(x)}{\partial x}$ of $a_i(x)$ with respect to each pixel of the input image (this is the actual gradient ascent step). - *modify the image* adding a small percentage of the gradient $\frac{\partial a_i(x)}{\partial x}$ and repeat the process until we get a *sufficiently high activation of the neuron*.

[!WARNING] There's no real difference between the gradient ascent and gradient descent process, since we can convert a minimization problem into a maximization problem by just negating the objective function. It's called gradient ascent only because we are trying to *increase* a value (which is the value of the activation of the neuron).



Example of visualization

A different approach

A different approach would be in using an input image and trying to understand which parts of the image are actually recognized by the network. To do that, we take the image, we take a particular internal layer, and what we do is trying to minimize the loss between the original image and the internal representation of the image that we are interested in. In this way, we are basically *synthesizing an image that is not distinguishable* from the original image in that specific layer of the network (meaning, they would produce the same activation).

Essentially, we are trying to understand the inner representation at some layer by generating an image indistinguishable from the original one.

The technique

- Goal: given an input image x_0 with an internal representation $\Theta_0 = \Theta(x_0)$, generate a different image x such that $\Theta(x) = \Theta_0$,
- Approach: via gradient ascent starting from a noise image. *Instead of optimizing towards a given category or the activation of a neuron, minimize the*

$$\underset{x}{\operatorname{argmin}} \underbrace{\ell(\Theta(x), \Theta_0)}_{\text{loss}} + \underbrace{\lambda \mathcal{R}(x)}_{\text{regularizer}}$$

distance from Θ_0 :

Obviously, it is much simpler to minimize this function when we are at a layer that is close to the start of the network, since this is when the result is much more similar to the starting image. - The more we traverse the network, the more the input becomes deconstructed and so it is more difficult to reconstruct.



Results

As we can see, the input becomes progressively fuzzier, and it seems that our network almost deconstructs the whole image.



Inceptionism

you've seen this shit for sure in some creepy youtube video. Essentially, it is image manipulation that injects inside the image the notions that we have just said, by applying the gradient descent techniques to particular layers of the network. [this is what asperti said, I know it is not quite clear but in the next paragraph it will be explained better]

Deep dreams

Initially intended to visualize what a deep neural network is seeing when it is looking in a given image (so it is the gradient descent techniques), it is now used as a procedural art form for making new form of psychedelic and abstract art.

The approach

- *train a network* for image classification
- *revert the network to slightly adjust* (via backpropagation) *the original image* to improve *activation of a specific neuron*.
- after enough reiterations, even imagery *initially devoid of the sought features will be incepted by them*, creating psychedelic and surreal effects;
- the generated images take advantage by strong regularizers privileging inputs that have statistics similar to natural images, like e.g. correlations between neighboring pixels (texture).

Enhancing content Instead of prescribing which feature we want to amplify, we can also fix a layer and enhance whatever it detected. Each layer of the network deals with features at a different level of abstraction. Lower layers will produce strokes or simple ornament-like patterns, because those layers are sensitive to basic features such as edges and their orientations.

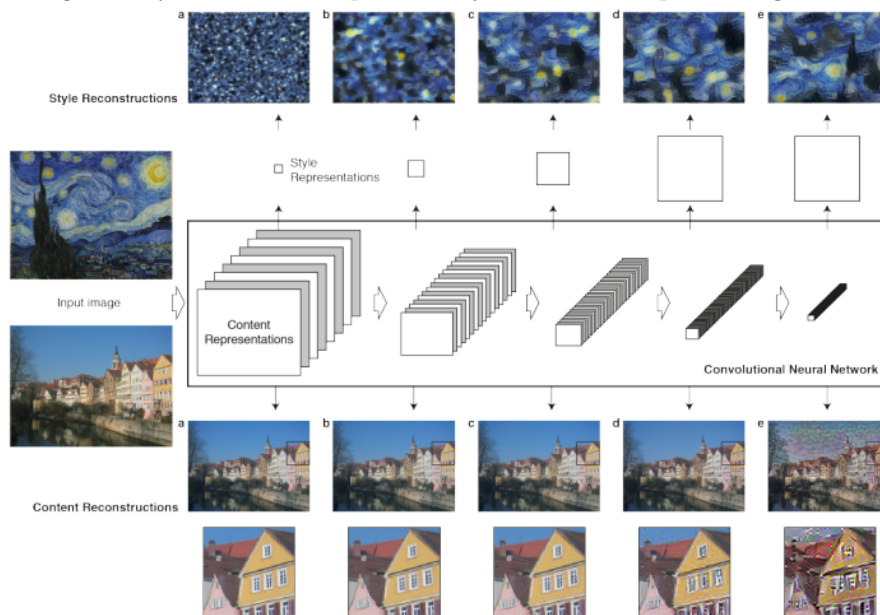
Style transfer

The gradient ascent technique can also be adapted to superimpose a *specific style*



to a given content:

To capture the style of another image, we can use techniques that come from the standard image processing field. In particular, we add a *feature space* on top of the original CNN representations which *computes correlations* between *the different features maps* (channels) at each given layer. A technique already used to compute image textures.



Gram Matrix We know that at layer l : - an image is encoded with D^l distinct feature maps F_d^l - each of size M^l (width times height). - $F_{d,x}^l$ is the activation of the filter d at position x at layer l .

Feature correlations for the given image are given by the **Gram matrix** $G^l \in R^{D^l \times D^l}$ where G_{d_1,d_2}^l is the dot product between the feature maps $F_{d_1}^l$ and $F_{d_2}^l$ at layer l :

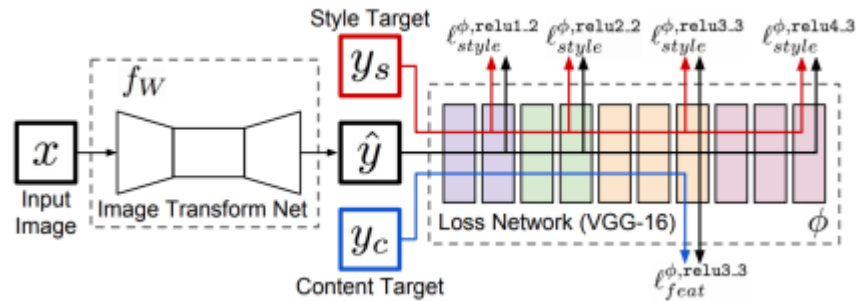
$$G_{d_1,d_2}^l = F_{d_1}^l \cdot F_{d_2}^l = \sum_k F_{d_1,k}^l \cdot F_{d_2,k}^l$$

The Gram matrix represents the internal representation of the image from the POV of style.

Combine style and content Content and style are separable, and in fact are two different inputs of the model.

Different combinations varying the reconstruction layer (rows) and the relevance ratio between style and content (columns) -> meaning, we can privilege either *style* or *content* (of the original image).

Variants and improvements (original work of this topic) The original work did not use the gradient descent technique, instead it use something like



this:

We can see that the loss function is represented as a network (in particular the *pre-trained* network VGG-16).

The input image x could be some *random noise*, that is fed inside an *image transformation network*, which in turn is trained to transform input images into output images. The image transform network is the only part of this model that involves some training.

Recap: possible applications of gradient ascent

- if the loss corresponds to the activation of a specific neuron (or a specific layer) we may *try generate images that cause a strong activation of it*, hence explaining the role of the neuron inside the network (what neurons see of the world)

- if **the loss is the distance**, in the latent space, from *the internal representation of a given image*, we may *try to generate other images with the same internal representation* (hence explaining what features have been captured in the internal representation)
- if **the loss is the similarity to a given texture**, we may try to *inject stylistic information* in the input image
- what *if the loss is the distance from a target category in a classification network?* Can we hope to automatically synthesize images belonging to that category? (or at least having distinctive features of that category?)

How to fool a NN

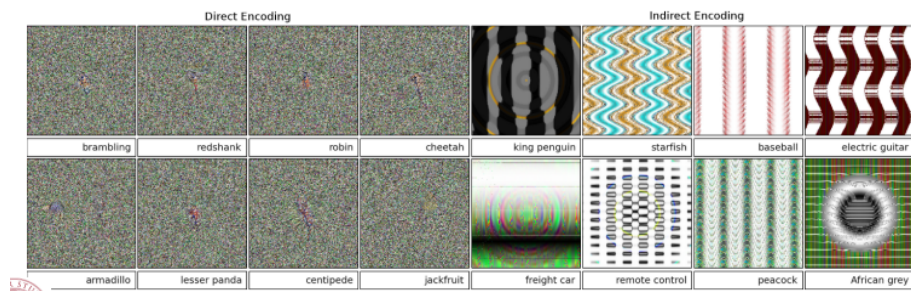
Since we have many pixels, a tiny (imperceptible to humans!), consistent pertur-



bation of all of them is able to fool the classifier.

Adversarial attacks and NNs as black boxes

The previous technique, being based on gradient ascent, *requires the knowledge of the neural net* in order to fool it. We can do something similar using the network as a black box, for instance by means of *evolutionary techniques*. These evolutionary techniques create images optimized so that they generate high-confidence DNN predictions for each class in the dataset. The evolutionary approach is this: - start with a random population of images - alternately apply selection (keep best) and mutation (random perturbation/crossover)



As we can see in the image, they were able to produce not only “noisy” adversarial images, but also geometrical examples with high regularities (meaningful for humans). - In the *indirect encoding*, rather than modifying pixels of an image directly, we simply create images in a parametric way, using geometrical shapes. The algorithm then acts on the parameters of the shapes. The result is are more complex images that is not just simple noise. - Indirect encoding is much slower.