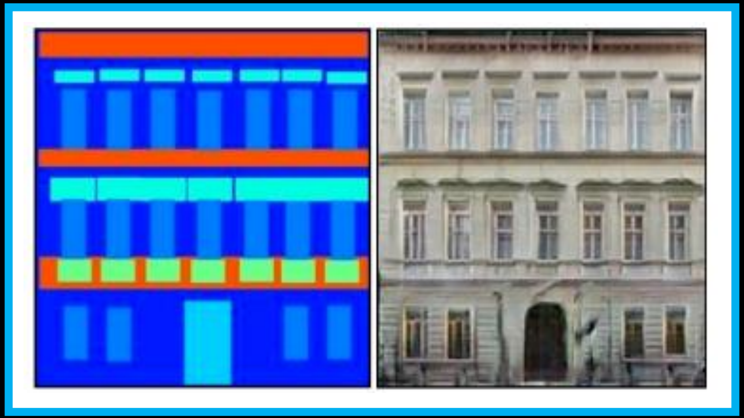
Exploring Machine Intelligence

Week 6, Generative Models II

~ Vítek Růžička

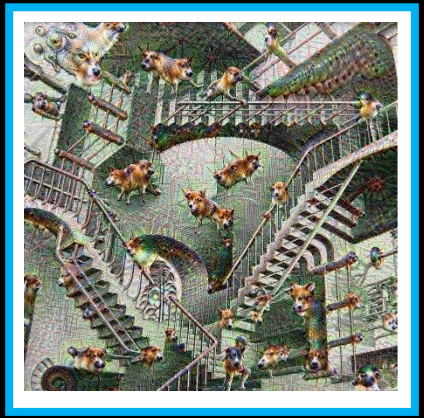
Motivation for today

Overview of additional machine learning techniques:



pix2pix

style transfer



deep dream

Today

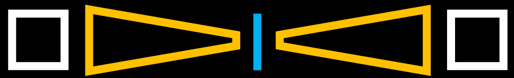
Overview of some additional Generative Models:

- **Pix2pix** and domain to domain transfer
- **Style transfer** technique
- **Deep Dream** technique

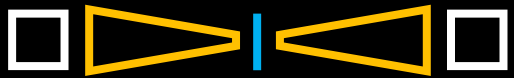
Larger focus on the practical session:

- Using **Progressively Growing GAN** – detailed instructions from data processing to model training

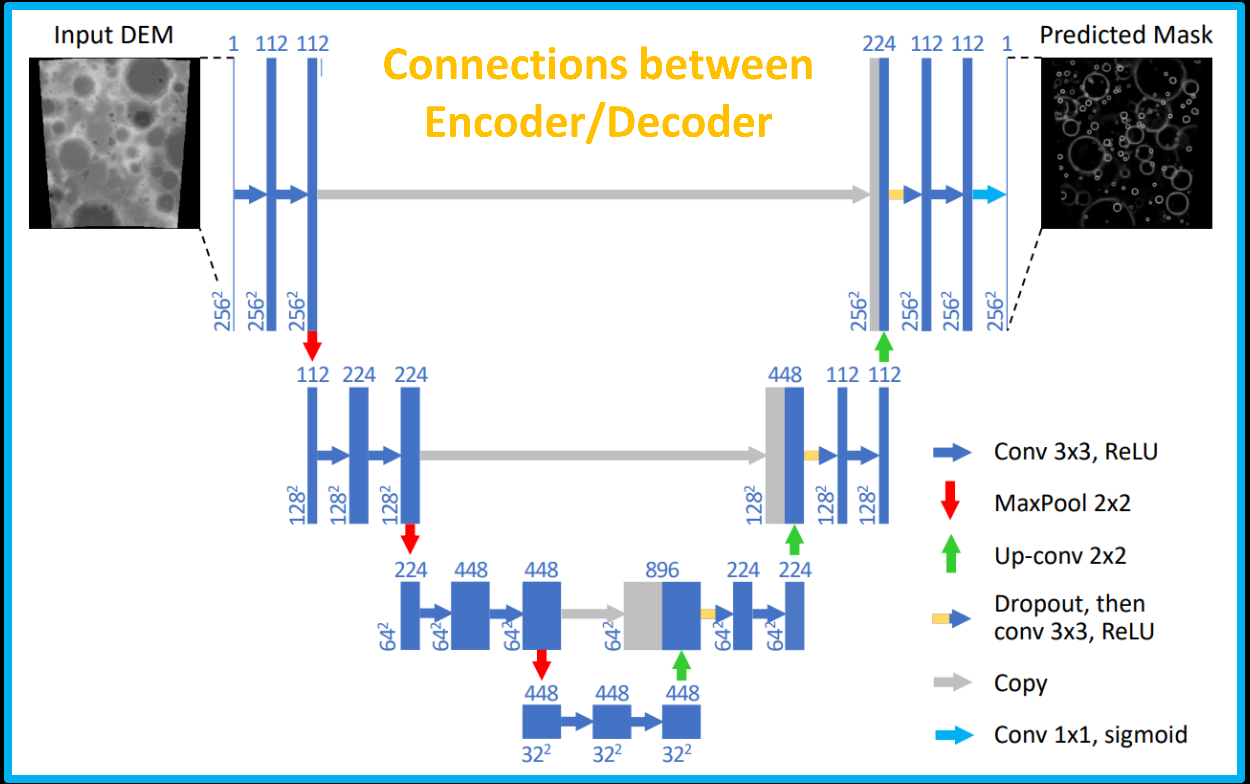
Domain to Domain

- We have seen few models working with images on outputs and inputs (for example AutoEncoders). Similar architectures can be also used to model **relation between two domains** of data. 

Domain to Domain

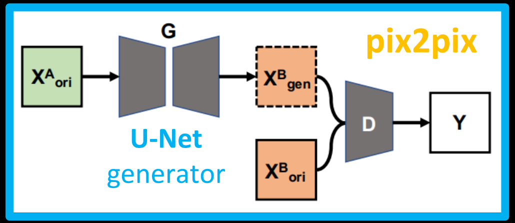
- We have seen few models working with images on outputs and inputs (for example AutoEncoders). Similar architectures can be also used to model **relation between two domains** of data. 

- Let's look at a predecessor of this idea, the **U-Net model**:
  - Paper with U-Net applied on the task of lunar crater identification

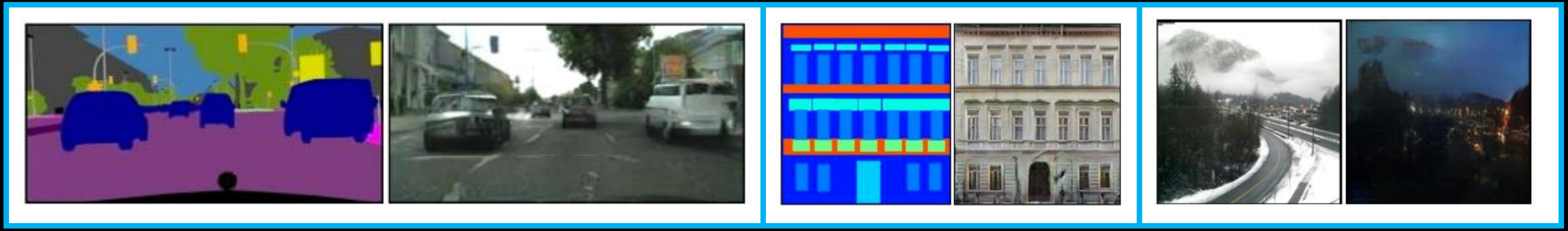


(PS: U-Net is from 2015, this [paper](#) from 2018)

Pix2Pix

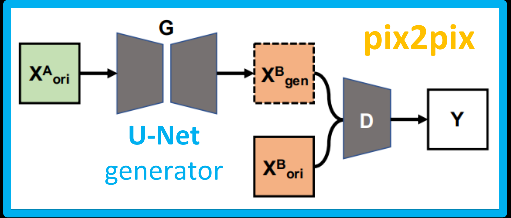


- General purpose **image-to-image translation:**
  - From their paper: *"Many problems in image processing, computer graphics, and computer vision can be posed as "translating" an input image into a corresponding output image."*
  - Translating without specifically defining the rules – data-driven translating between two domains by showing **paired** examples:
    - [Image from A, Corresponding Image from B] * N samples



(PS: there is also an un-paired version called CycleGAN = images from two domains, but they don't have to correspond)

Pix2Pix



- General purpose **image-to-image translation:**

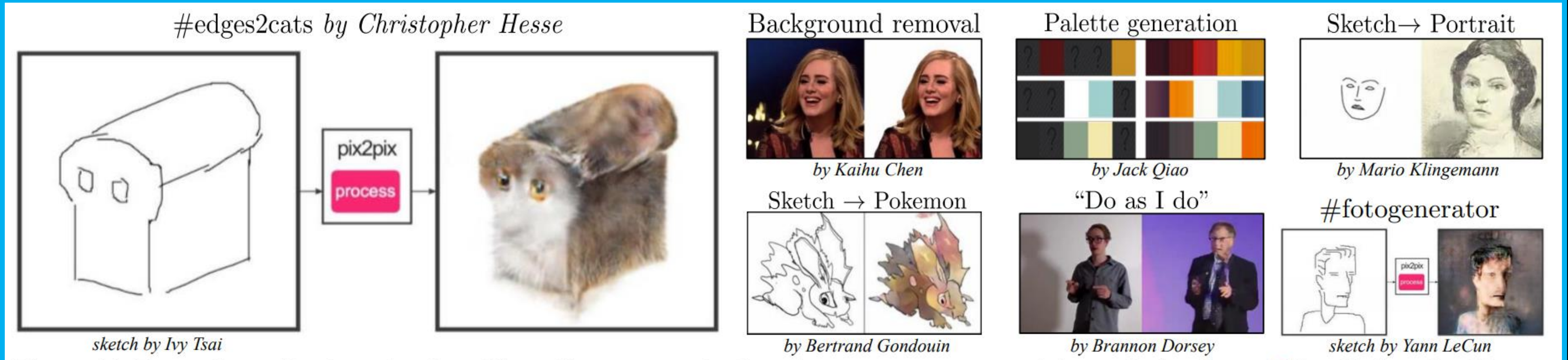


Figure 11: Example applications developed by online community based on our `pix2pix` codebase: *#edges2cats* [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch→ Pokemon* [1] by Bertrand Gondouin, *"Do As I Do" pose transfer* [2] by Brannon Dorsey, and *#fotogenerator* by Bosman et al. [4].

Pix2Pix

- **Online demos**: affinelayer.com/pixsrv/
- **Colab notebooks**:
  - Training and using Pix2Pix on Colab: our repo / Demo1_pix2pix-keras-v2.ipynb
- **Papers**: pix2pix, pix2pixHD (with high. res.), vid2vid (with frame to frame consistency)

Style Transfer

- Previously we saw, how a Deep Convolutional network separates where it saves *high-level* and *low-level* representations (this is due to the Conv->Pool combo – each convolutional layer serves as an image-filter).

Style Transfer

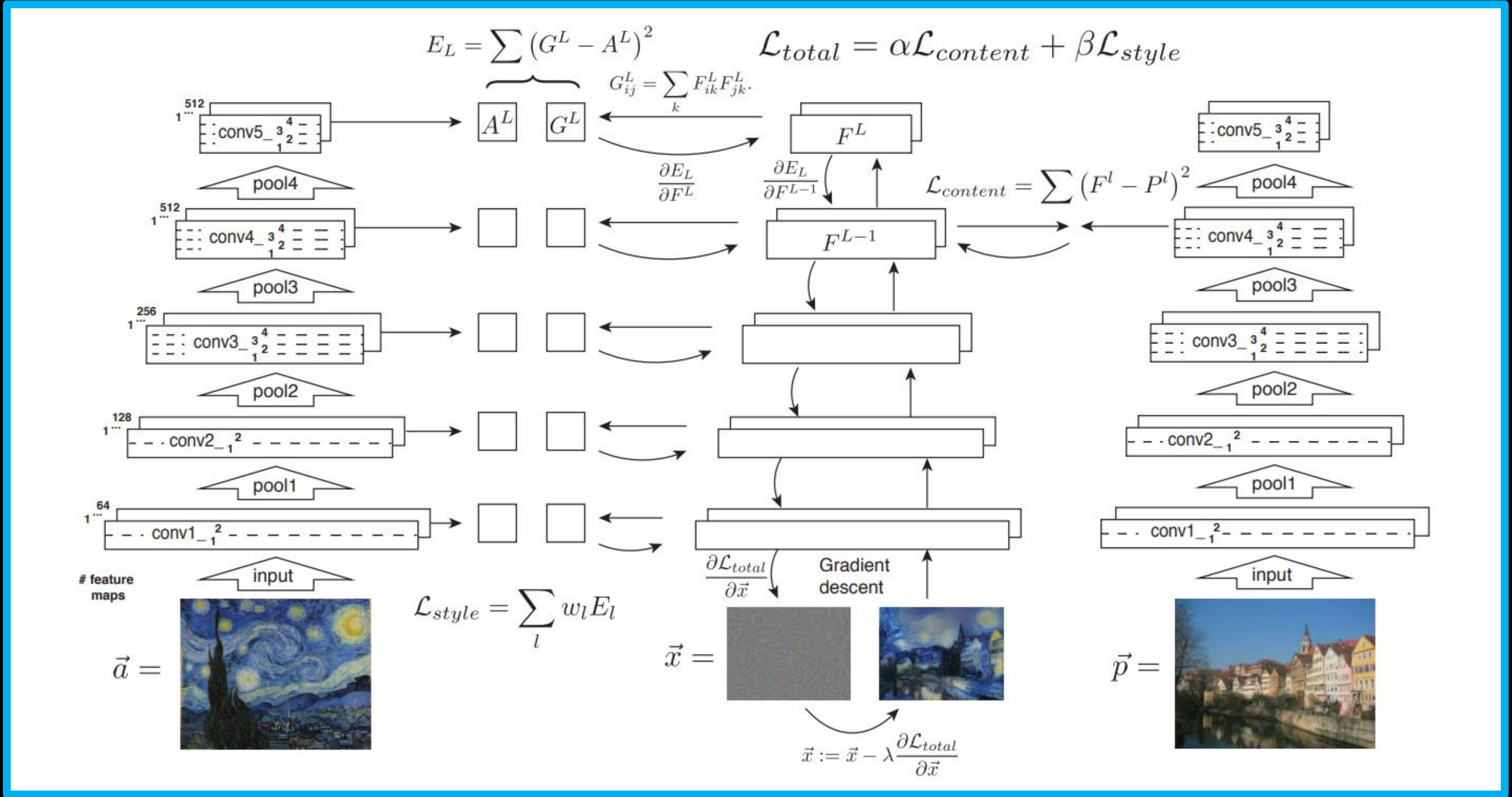
- Previously we saw, how a Deep Convolutional network separates where it saves *high-level* and *low-level* representations (this is due to the Conv->Pool combo – each convolutional layer serves as an image-filter).
  - **High-level information** – contents of the whole image – **image content**
    - Encode one image extracting its content information, the feature responses in deeper layers of the network

Style Transfer

- Previously we saw, how a Deep Convolutional network separates where it saves *high-level* and *low-level* representations (this is due to the Conv->Pool combo – each convolutional layer serves as an image-filter).
  - **High-level information** – contents of the whole image – **image content**
    - Encode one image extracting its content information, the feature responses in deeper layers of the network
  - **Low-level information** – details used inside the image, texture – **image style**
    - Encode another image extracting the style information, feature response alongside a selection of layers

(PS: One possible similarity if the content of low and high frequencies in music converted to spectrograms)

Style Transfer



Style Transfer

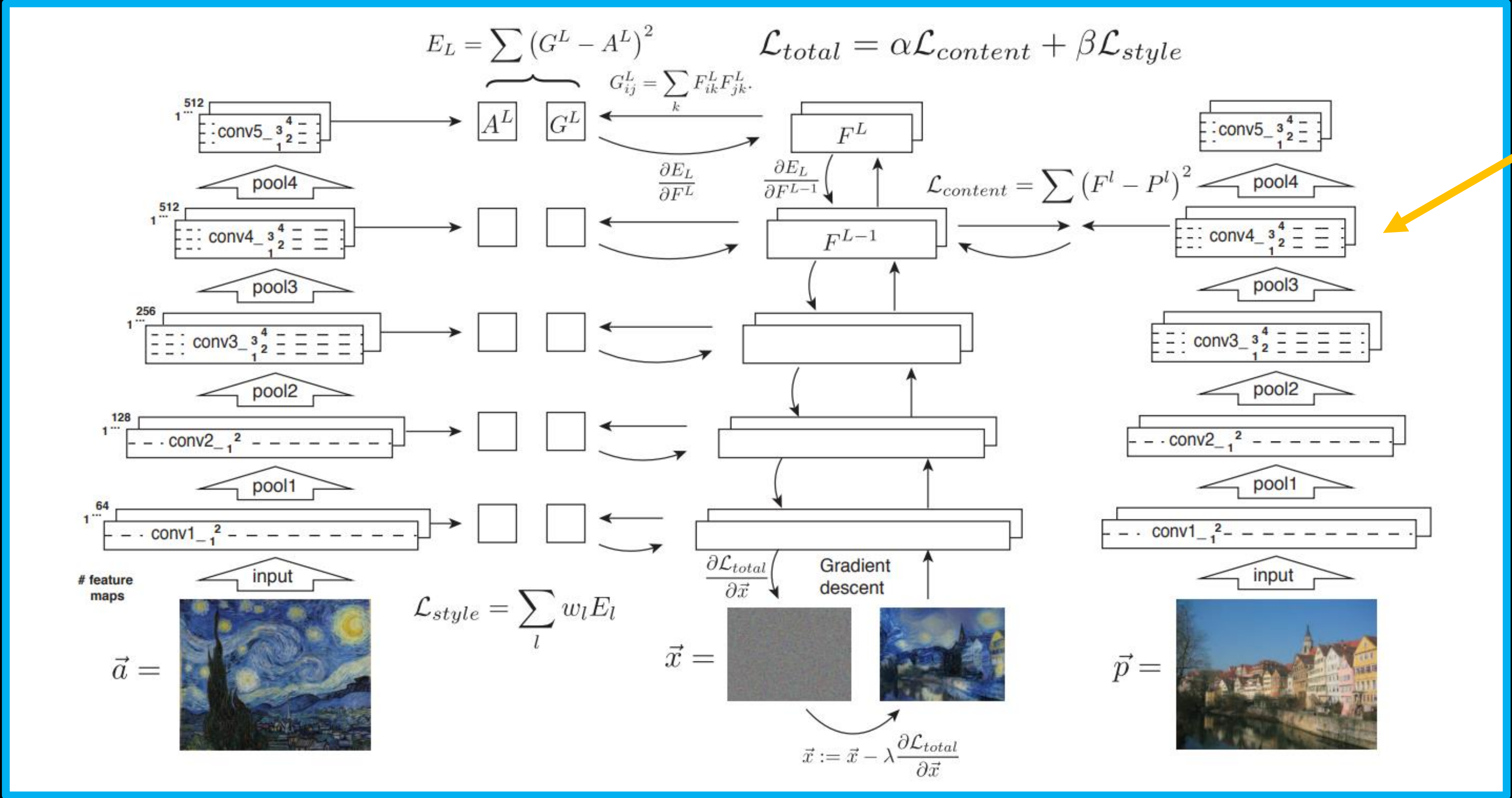


Image content = the feature responses in deeper layers of the VGG network

Style Transfer

Image style = feature response alongside a selection of layers

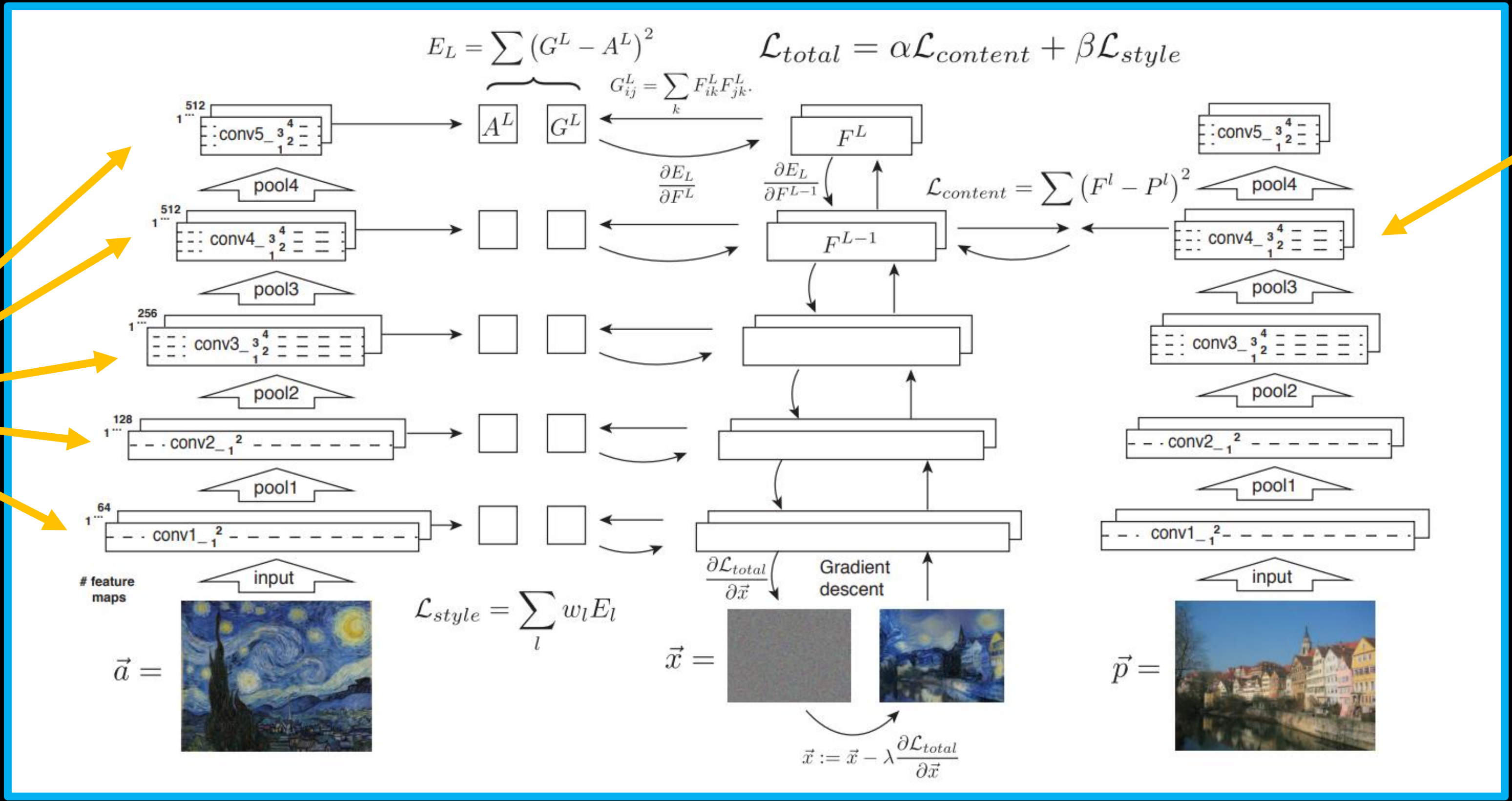


Image content = the feature responses in deeper layers of the VGG network

Style Transfer

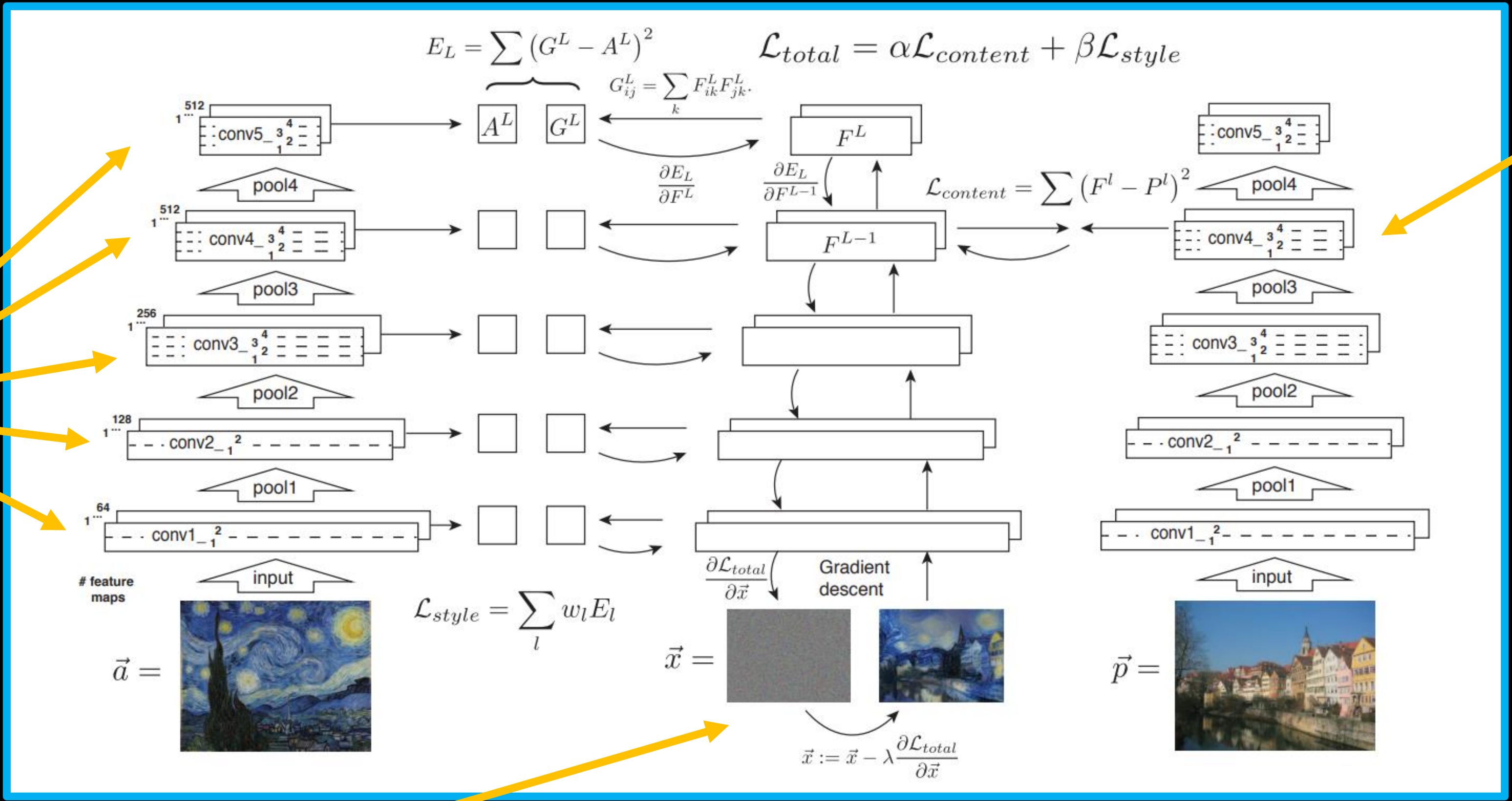


Image style = feature response alongside a selection of layers

Image content = the feature responses in deeper layers of the VGG network

Optimizing random noise image to have the same responses as those features we saved.

- Iteratively we will create a new image which has the style responses similar to our encoded style features + content responses similar to our encoded content features

Style Transfer

Image style = feature response alongside a selection of layers

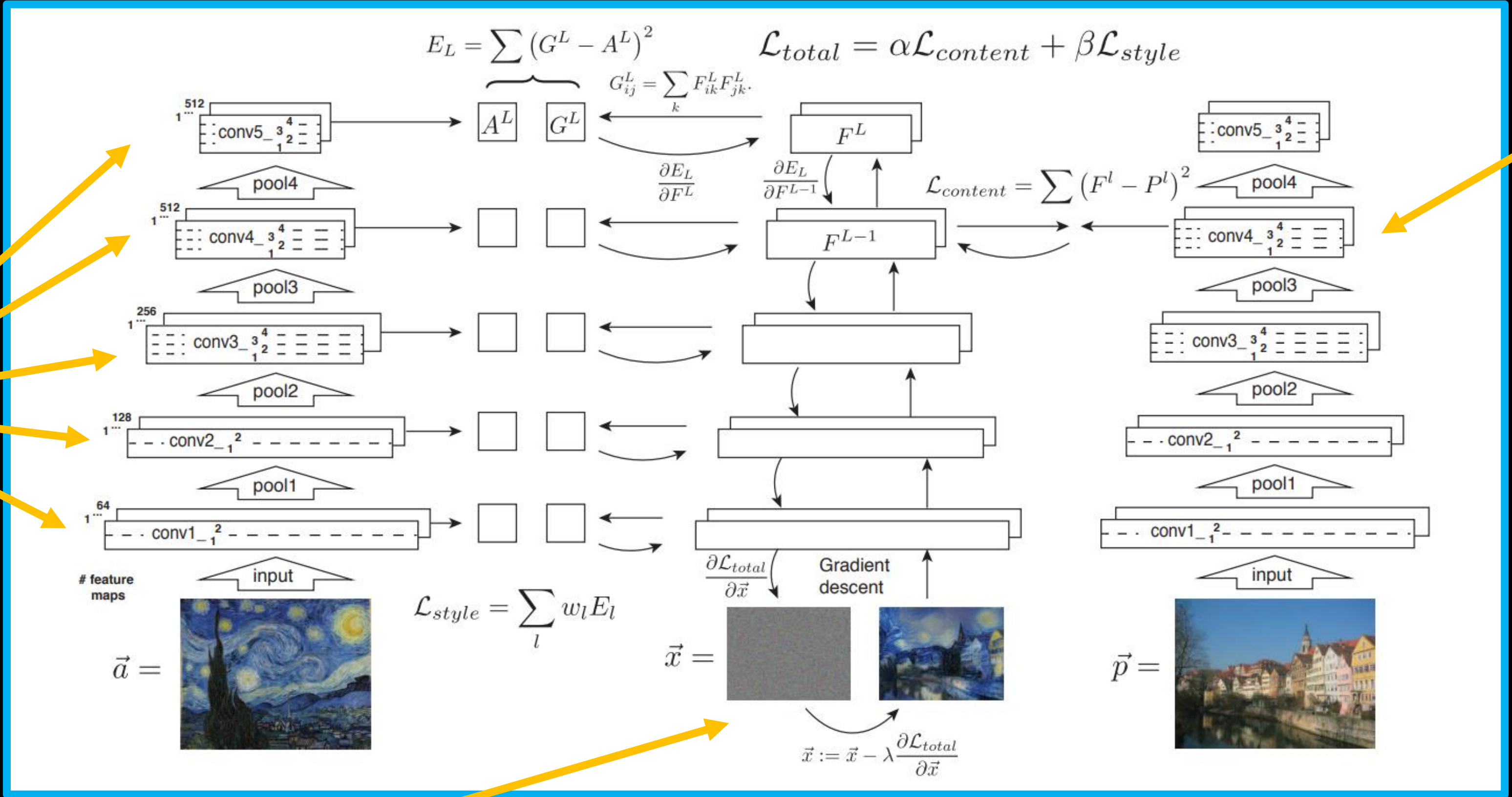


Image content = the feature responses in deeper layers of the VGG network

Optimizing random noise image to have the same responses as those features we saved.

- Iteratively we will create a new image which has the style responses similar to our encoded style features + content responses similar to our encoded content features

This is relatively slow (iterative optimization of the input image), later papers sped it up by using an image2image translation method with feed-forward networks.

Style Transfer

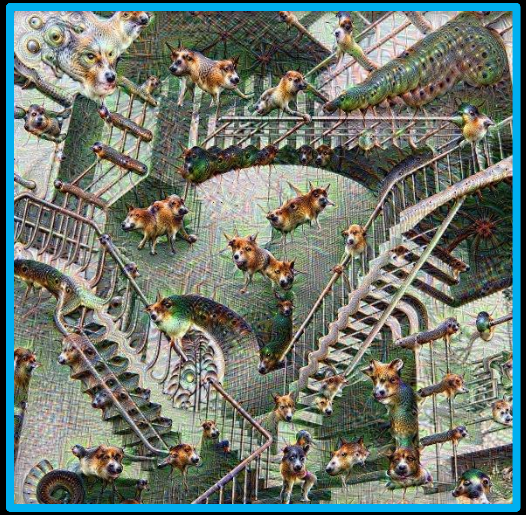
Style Transfer

- **Online demos**: deepart.io/latest/
- **Colab notebooks**:
  - Basic style transfer (with arbitrary images): ArtML / style_transfer_keras.ipynb
  - Fast style transfer (with pretrained styles): ArtML / fast-style-transfer

- **Papers**: style transfer (2015), fast style transfer (2016)

Deep Dream

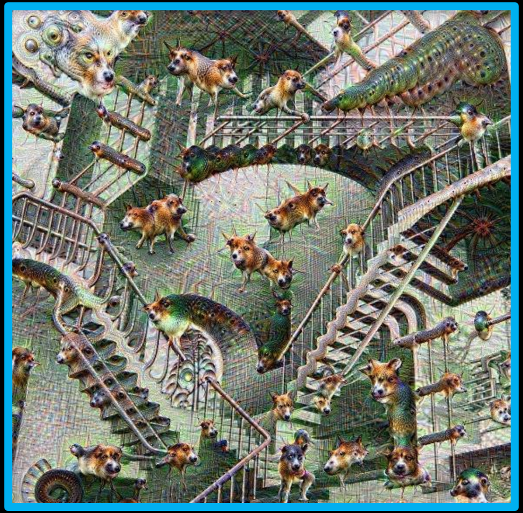
- Deep dream is a method which was originally used to visualize what a network has learned. It works on image optimization principle (as did the first version of Style Transfer).



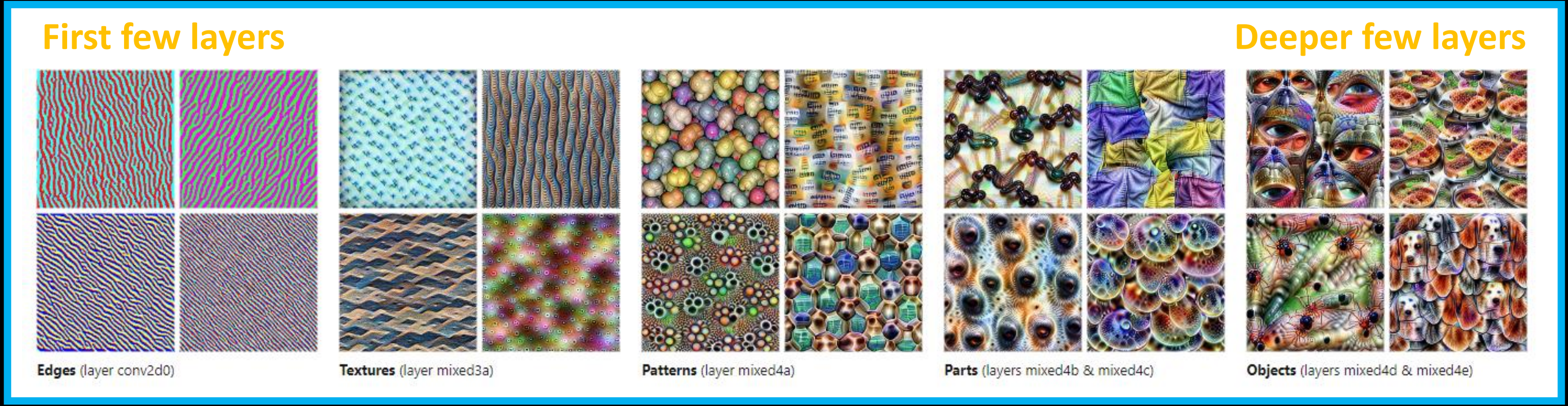
(PS: image source, features viz., illustrational video)

Deep Dream

- Deep dream is a method which was originally used to visualize what a network has learned. It works on image optimization principle (as did the first version of Style Transfer).



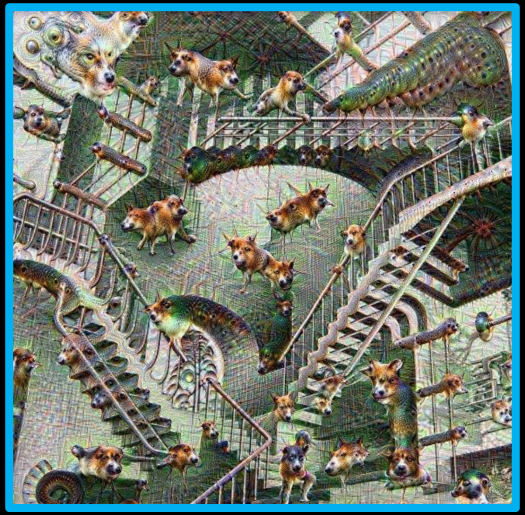
Using **Convolutional network** (GoogLeNet) trained for classification on ImageNet:



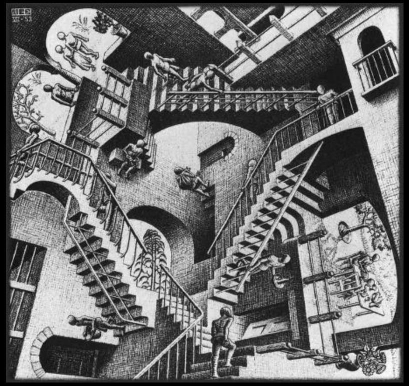
(PS: image source, features viz., illustrational video)

Deep Dream

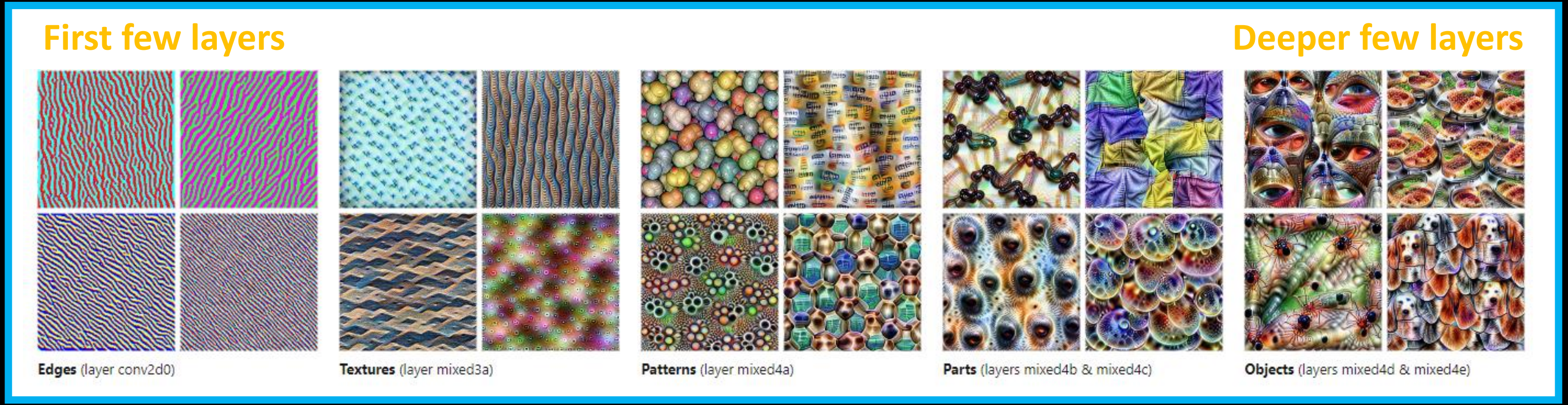
- Deep dream is a method which was originally used to visualize what a network has learned. It works on image optimization principle (as did the first version of Style Transfer).



Change this **original image** so that it activates a **selected feature** with the highest possible force!



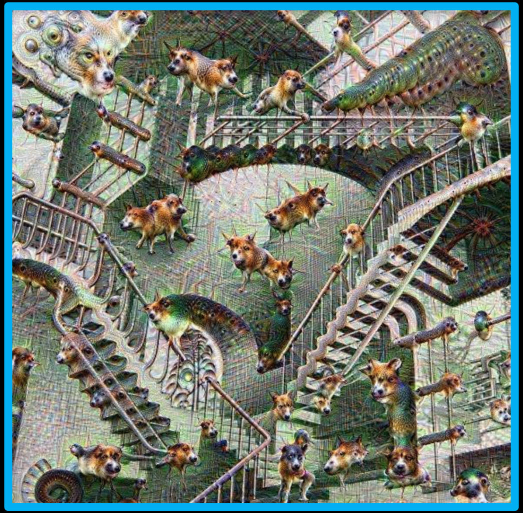
Using **Convolutional network** (GoogLeNet) trained for classification on ImageNet:



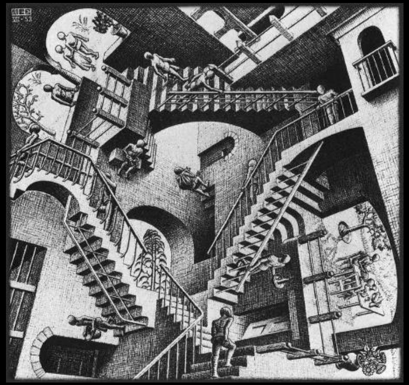
(PS: image source, features viz., illustrational video)

Deep Dream

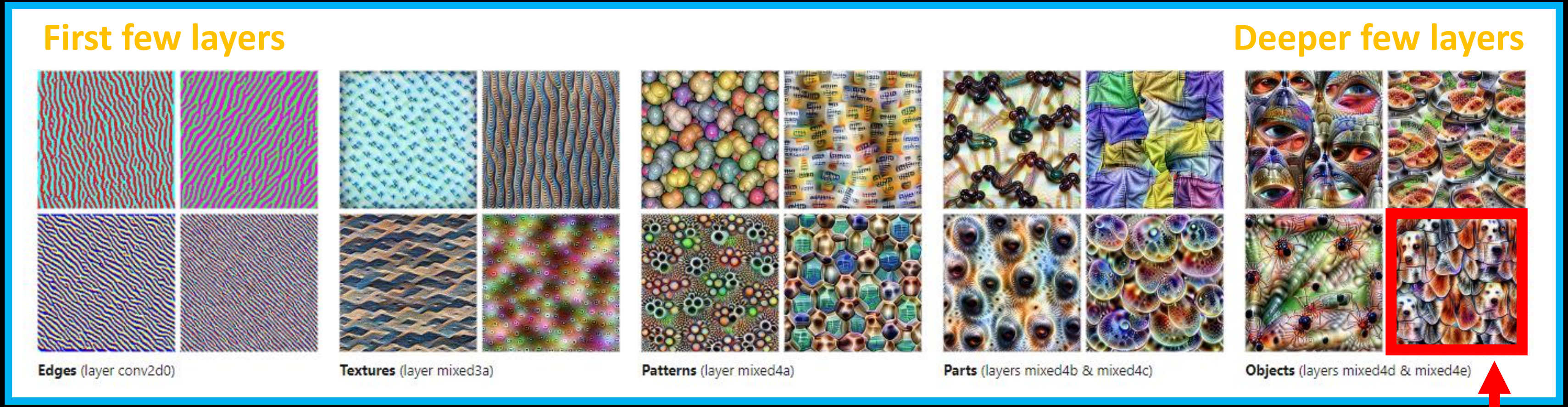
- Deep dream is a method which was originally used to visualize what a network has learned. It works on image optimization principle (as did the first version of Style Transfer).



Change this **original image** so that it activates a **selected feature** with the highest possible force!



Using **Convolutional network** (GoogLeNet) trained for classification on ImageNet:

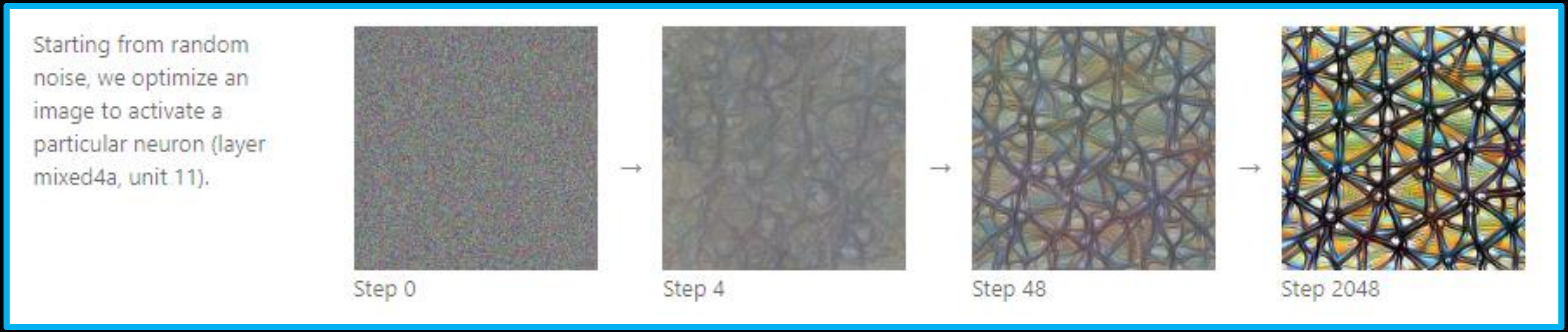


selected feature

(PS: image source, features viz., illustrational video)

Deep Dream

Iterations:



(PS: image [source](), features [viz.](), illustrational [video]())

Deep Dream

- **Online demos**: dreamscopeapp.com *(not 100% sure if it's not a style transfer of deep dream like effect)*
- **Colab notebooks**:
  - Deep dream a photo: as a ML4A guide
  - Alternative code: ArtML / neural-synth-clustering-v2.ipynb
- **Reading**: distill.pub/2017/feature-visualization/

Practicum: Generative Models II.

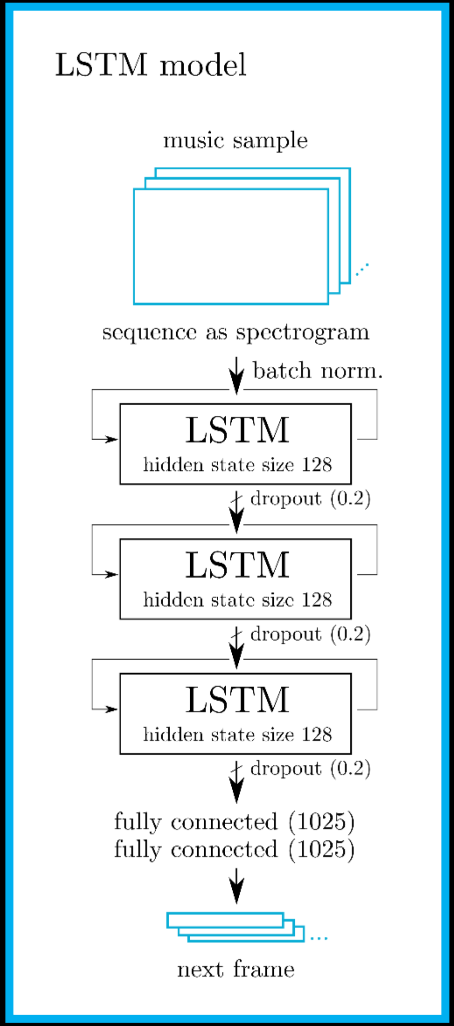
This week's focus will be in learning how to use **Progressive Growing GAN** with all practical steps included:

- data processing: 1_Dataset_Processing.ipynb
- model training and finally model inference: 2_Train_Progressive_GAN.ipynb

Next class

More generative models:

- **Sequential modelling**



Links and additional readings:

- **Bonus readings:**
  - Feature Visualization, Distill – blog
  - Sensory Optimization: Neural Networks as a Model for Understanding and Creating Art – paper
  - About Pix2Pix on ML4A – blog with code

The end