

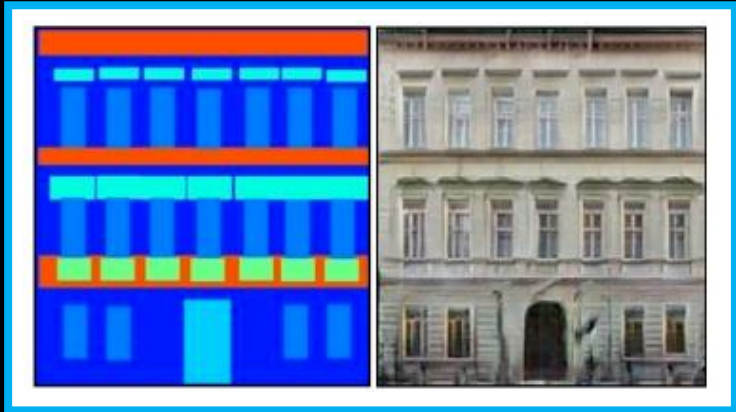
# AI for the Media

## Week 9, Domain 2 Domain



# Motivation for today

## Overview of additional machine learning techniques:

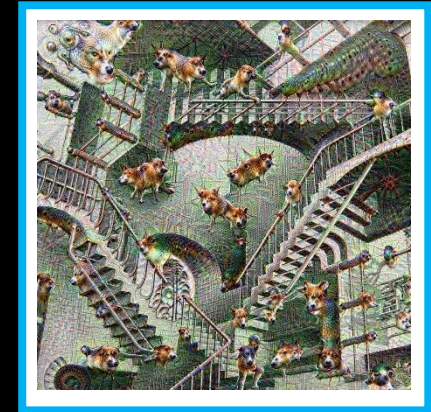


**pix2pix**

*& co*



**style transfer**



**deep dream**

*& adversarial  
attacks*

# Overview

## Domain 2 Domain Models (*pre-recorded lecture*):


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

## Practical session (*during the live session*):


- **Code**: Training a pix2pix model

**Note:** Today we will see a lot of models and methods, don't worry about understanding all of them, but focus on the concept of what is being learned by the models and how (by the special connectivity or by the special tricks in writing up the loss functions ... etc.)

# Domain to Domain

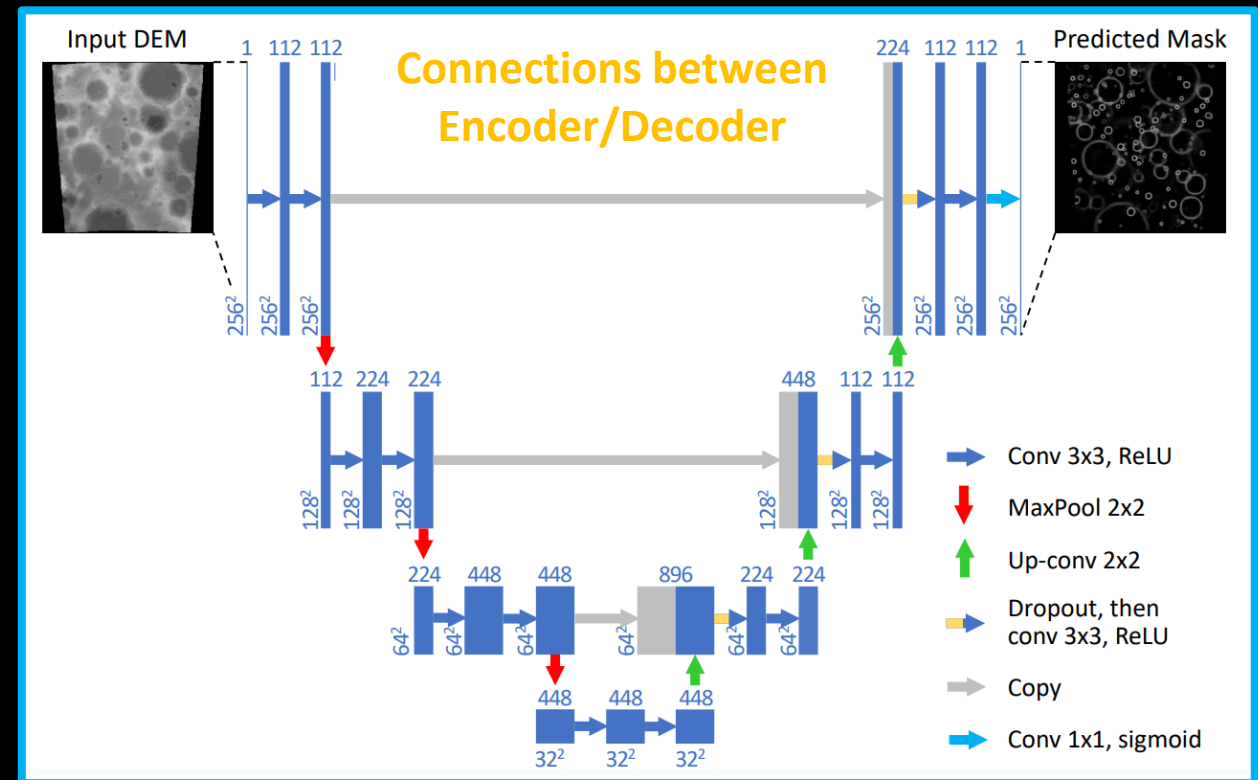
- We have seen few models working with images on outputs and inputs (for example [AutoEncoders](#)). Similar architectures can be also used to model **relation between two domains** of data. 

# Domain to Domain

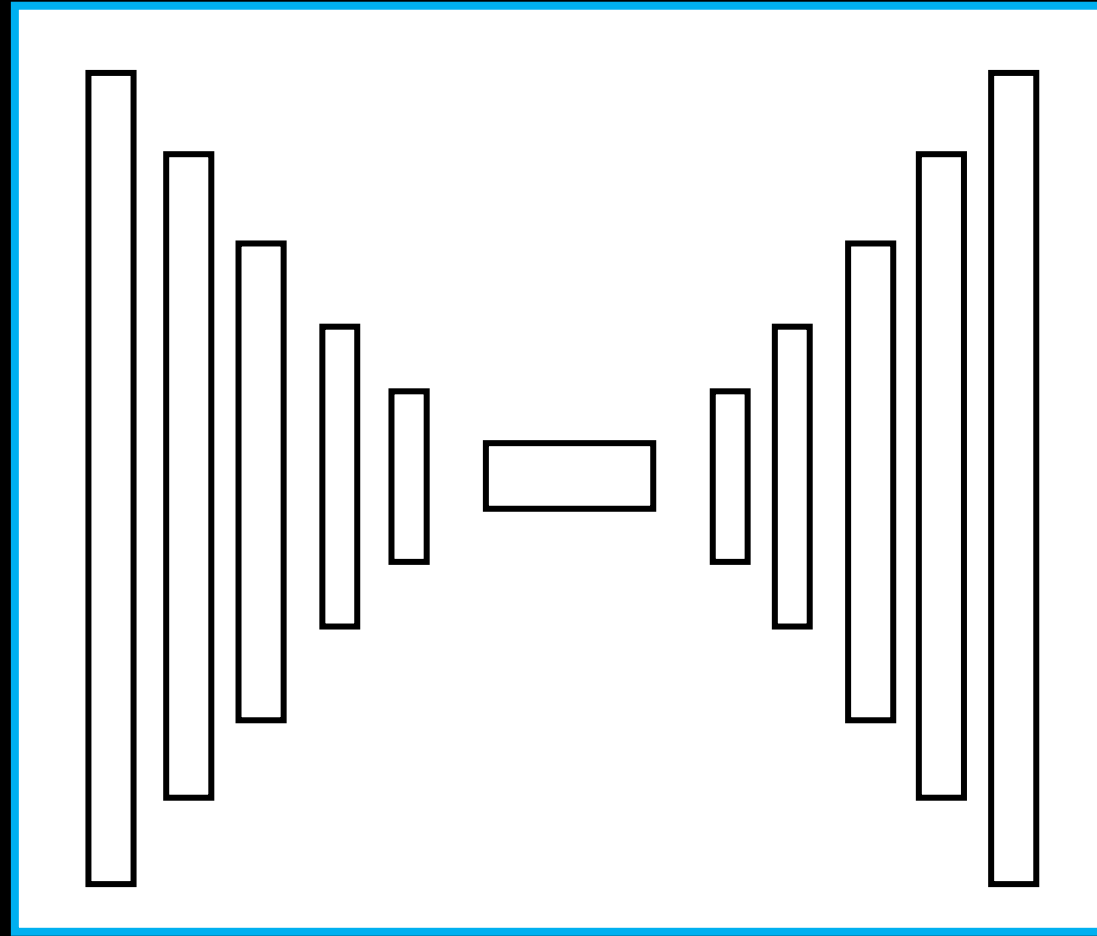
- We have seen few models working with images on outputs and inputs (for example [AutoEncoders](#)). Similar architectures can be also used to model **relation between two domains** of data. 

- 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

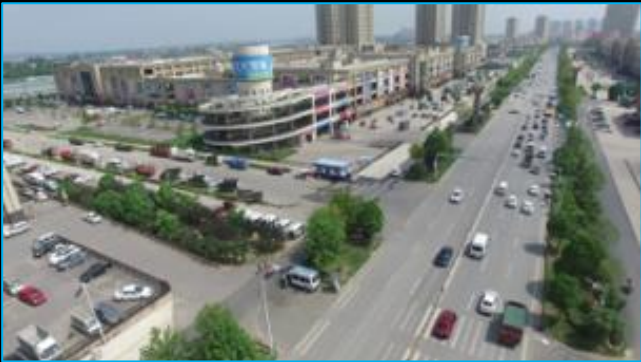
(PS: U-Net is from 2015, this [paper](#) from 2018)



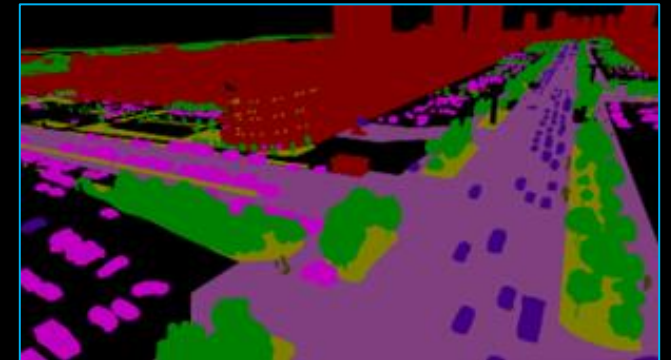
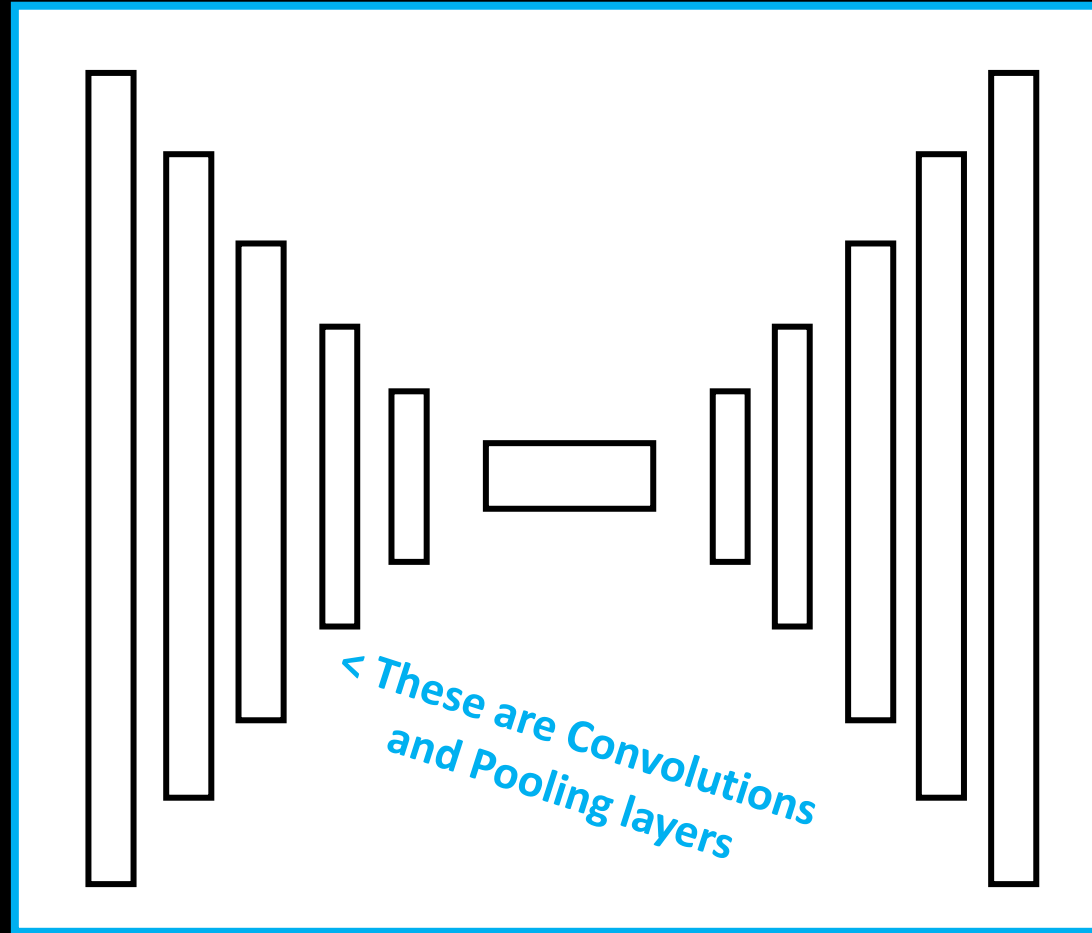
# U-Net



# U-Net



Images from one  
“domain”

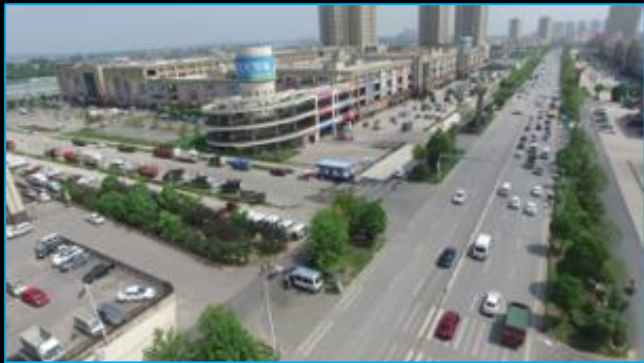


Paired images from  
another “domain”

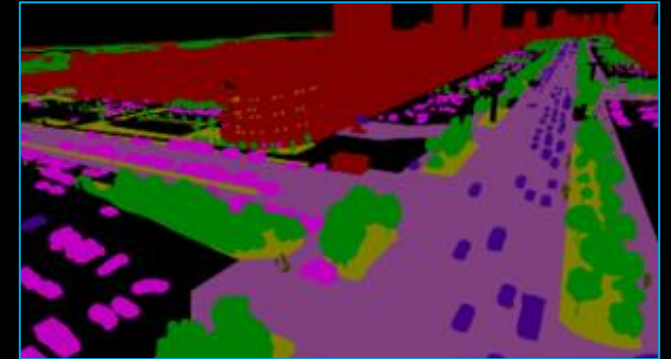
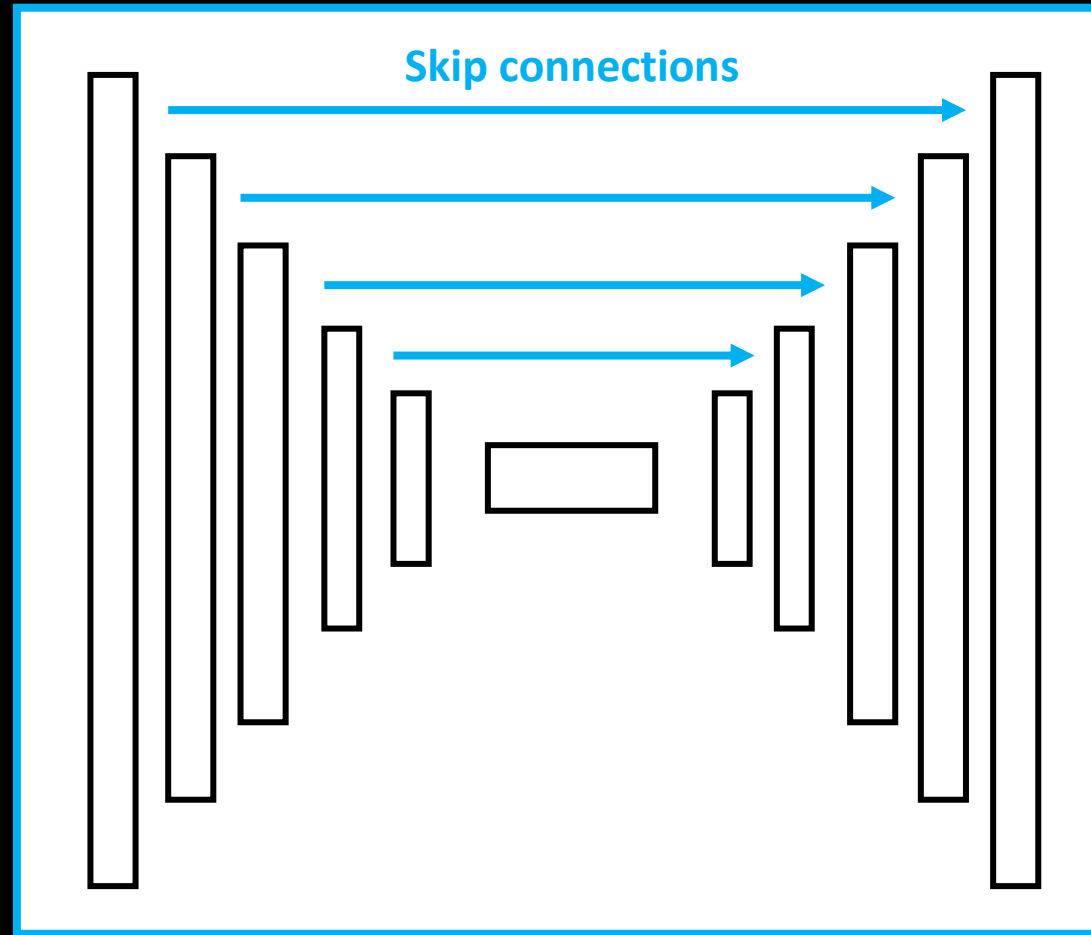
- Connects one image to another while going through a bottleneck



# U-Net



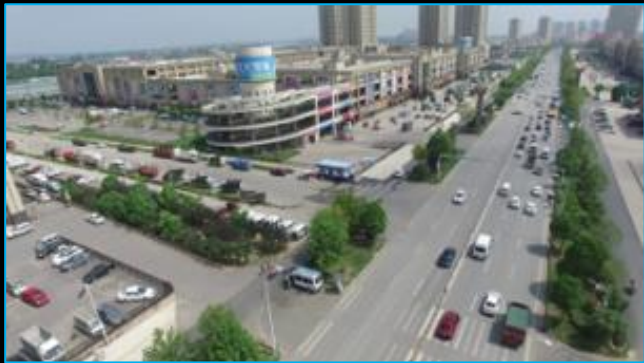
Images from one  
“domain”



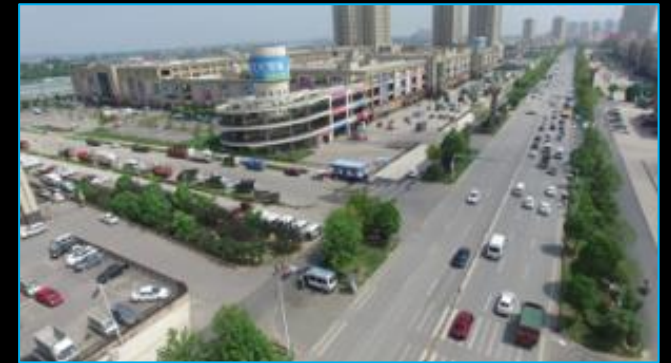
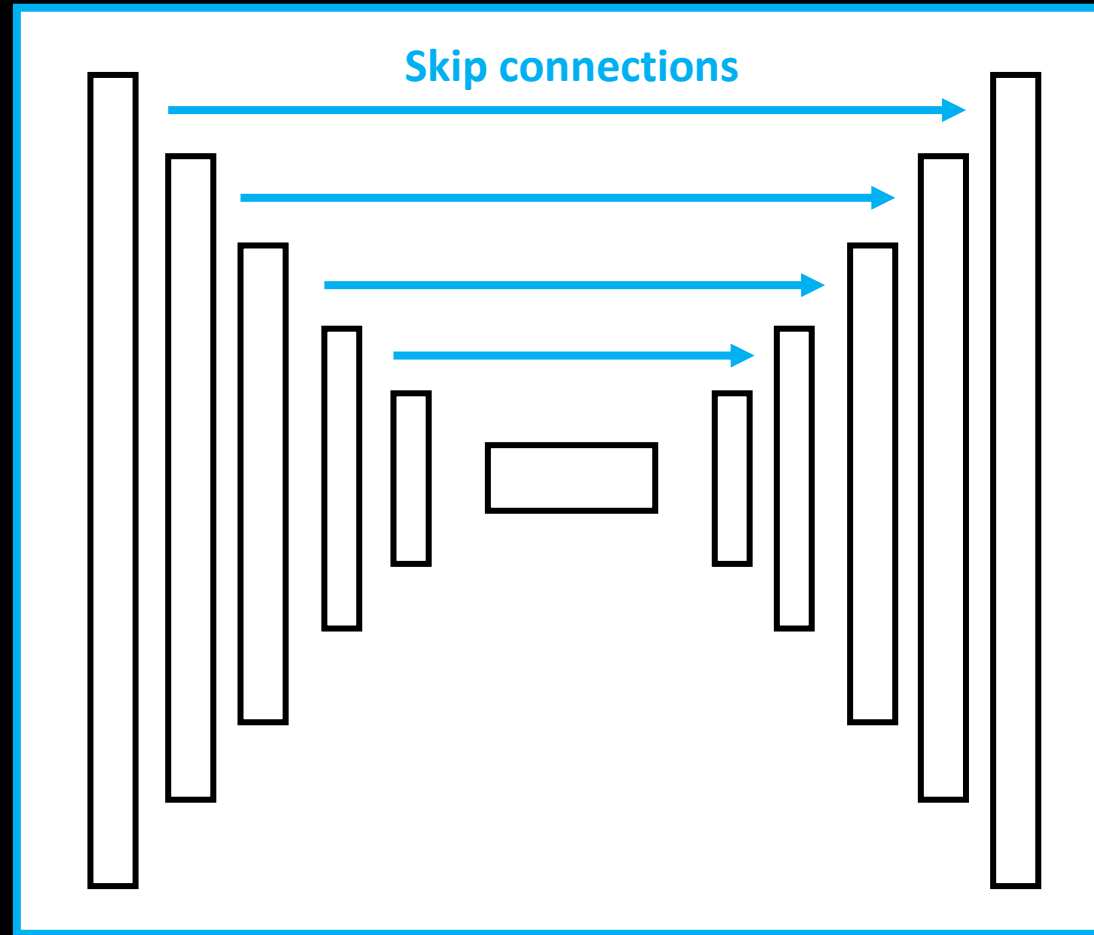
Paired images from  
another “domain”

- Additionally: “*skip connections*” that allow the information to flow

# U-Net



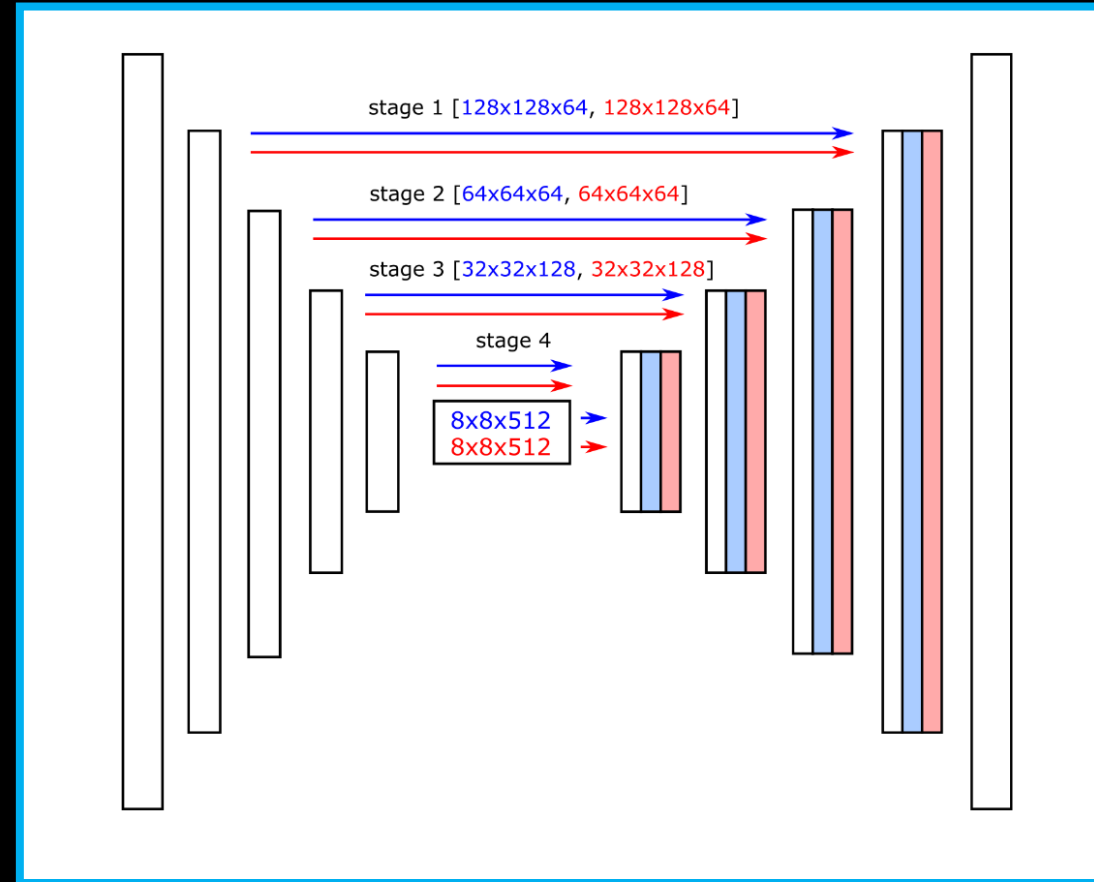
Images from one  
“domain”



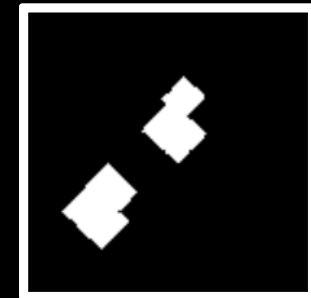
(if this was an  
autoencoder)

- PS: As an autoencoder, this model could very easily cheat ...

# U-Net in a real-world scenario



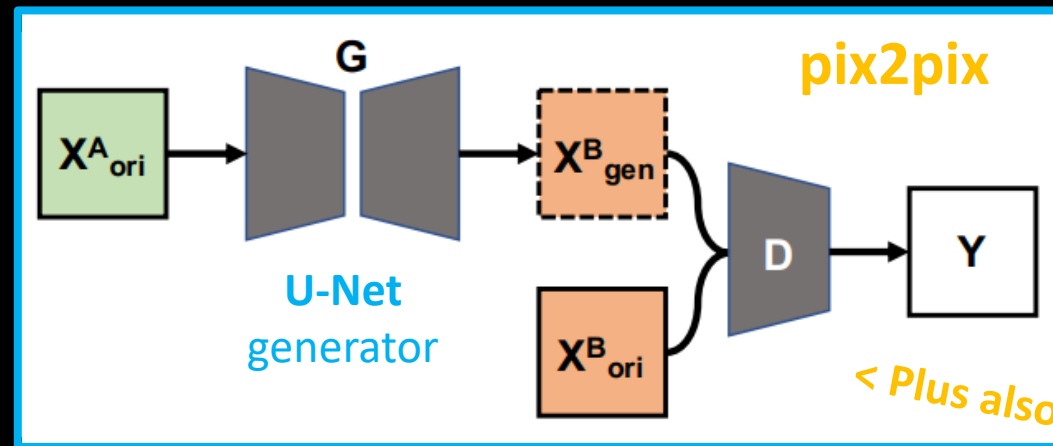
< Slightly more experimental setup  
– check Siamese U-Nets\*



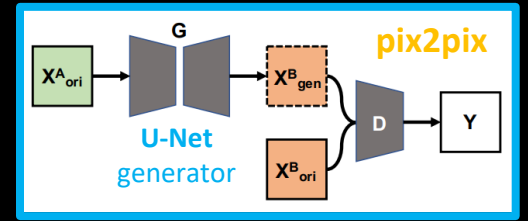
- Can be adapted to many tasks – for example this “change detection”, where it needed to learn *which kind of change* we care about

# Pix2Pix

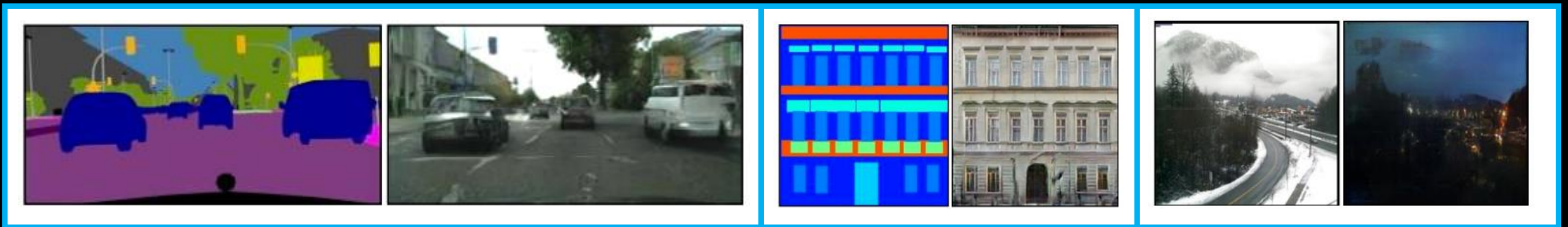
- General purpose **image-to-image translation**:
  - From their paper: “Many problems in image processing, computer graphics, and computer vision can be posed as “*translating*” an input image into a *corresponding output image*.”



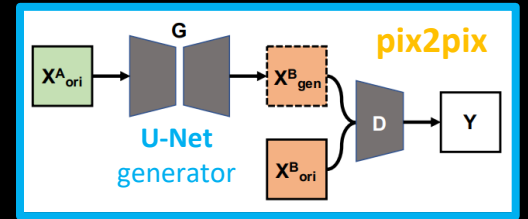
# Pix2Pix



- 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



# Pix2Pix



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

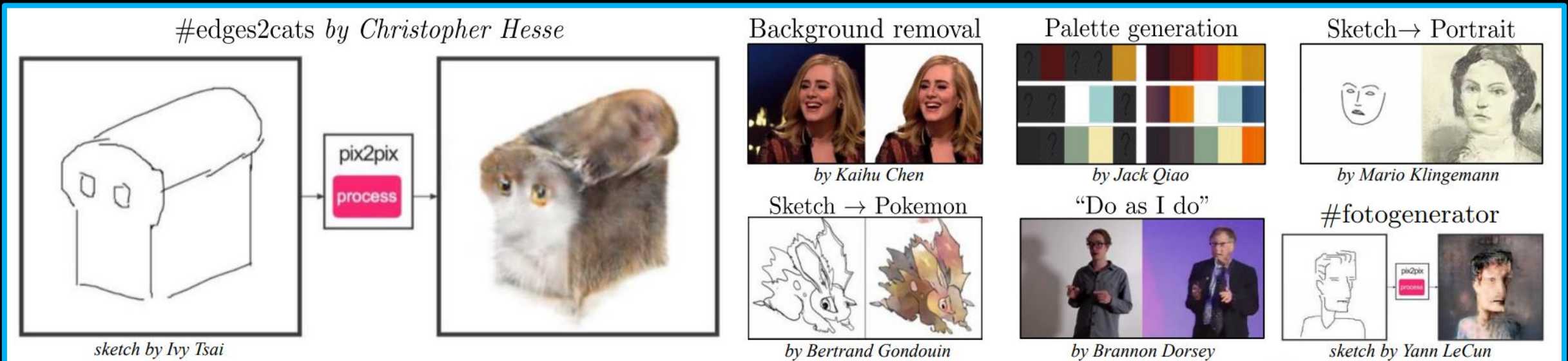
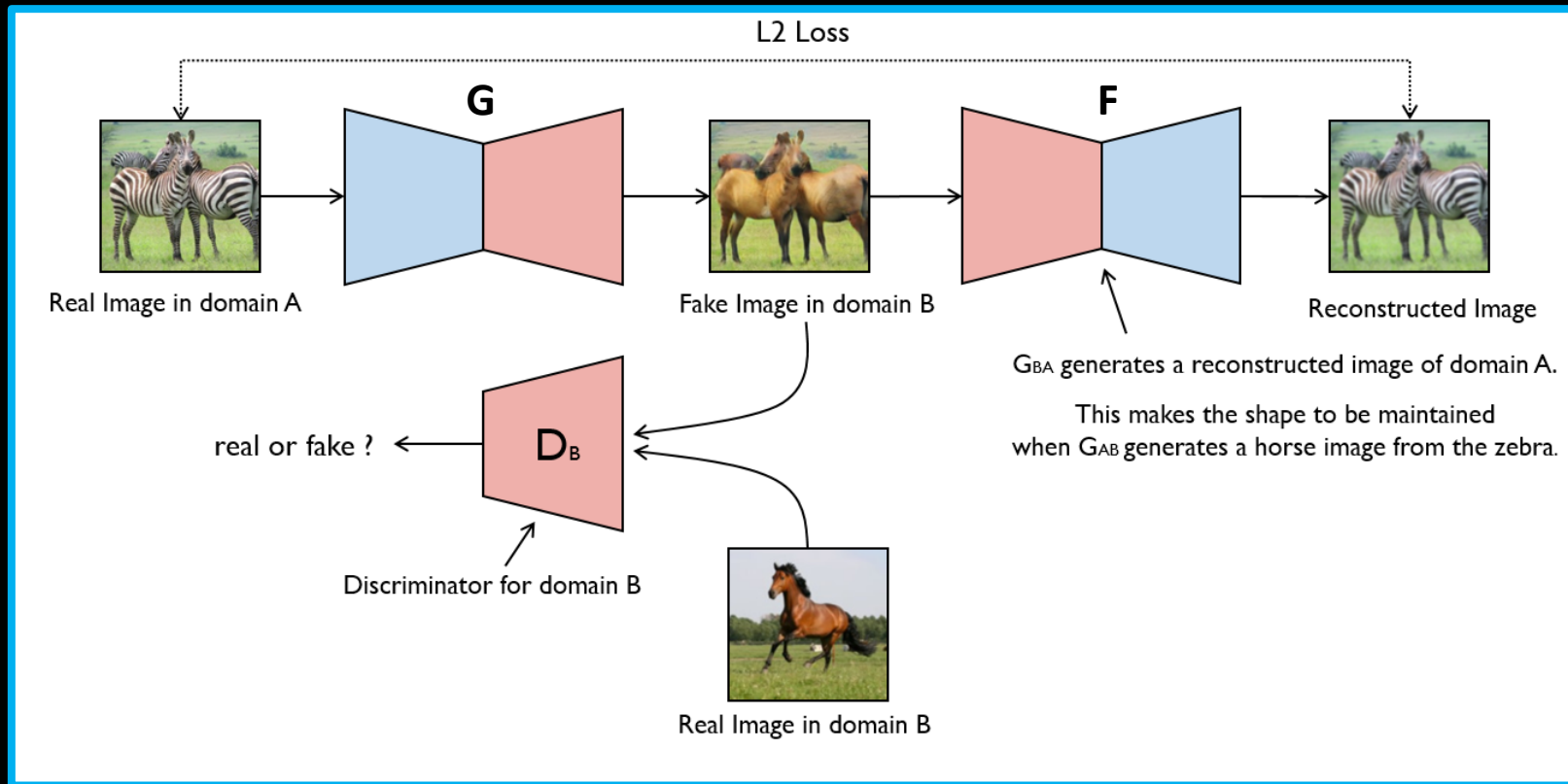


Figure 11: Example applications developed by online community based on our `pix2pix` codebase: `#edges2cats` [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch → Pokemon* [1] by Bertrand Gondouin,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

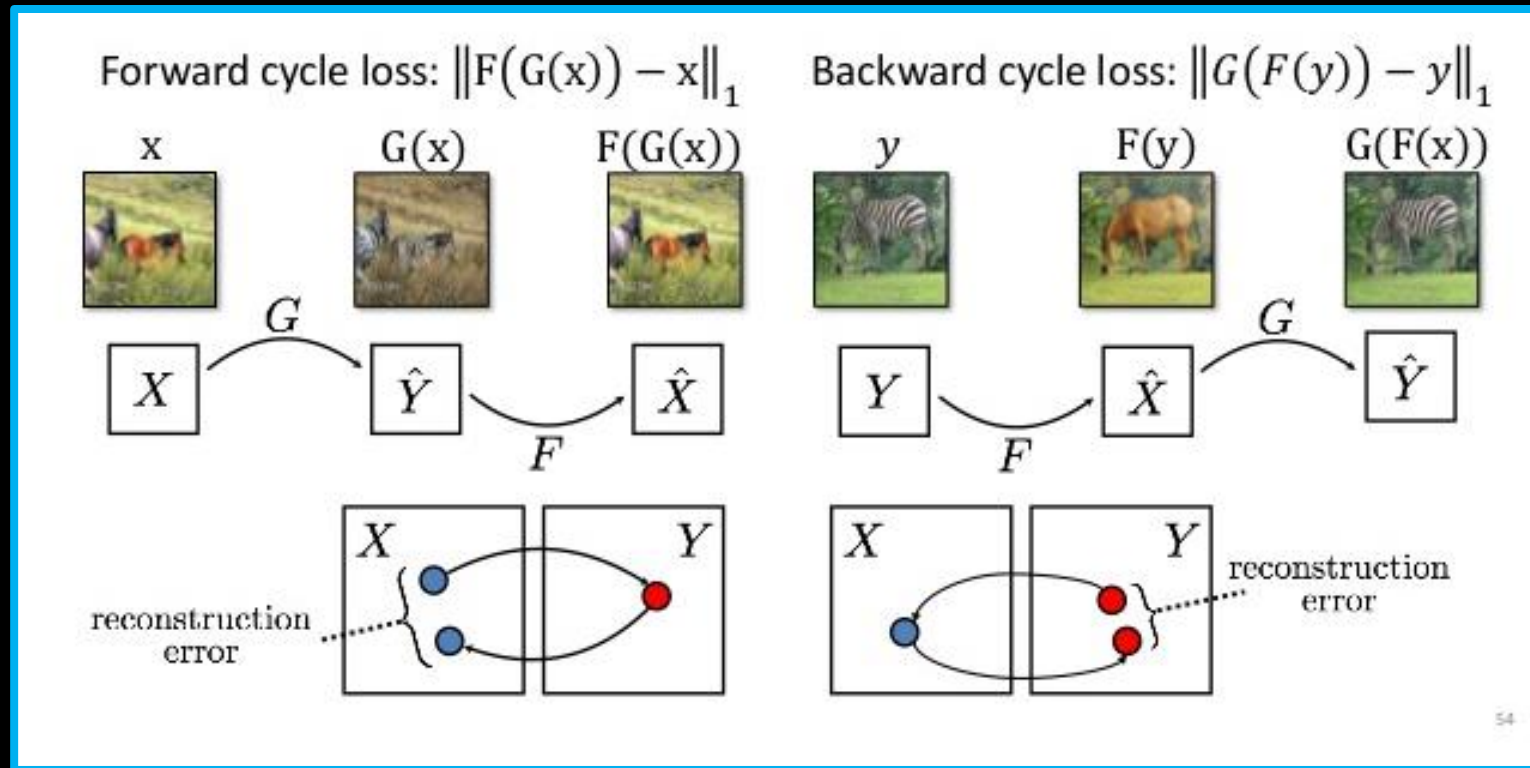


# CycleGAN, connectivity



- Setup with two generators going between two domains (**G** and **F**) and a special cyclic loss function. This model **doesn't require paired data** in the training datasets.

# CycleGAN, loss functions



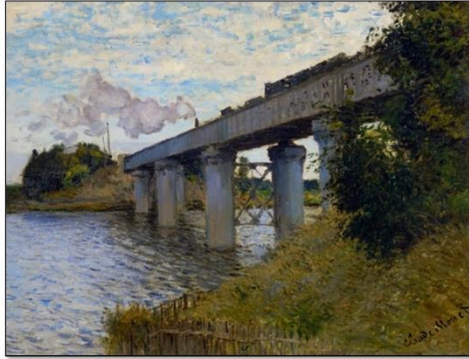
Check  
this  
[video](#)

- And a **cyclic loss function** which looks at two types of errors ^



# CycleGAN examples

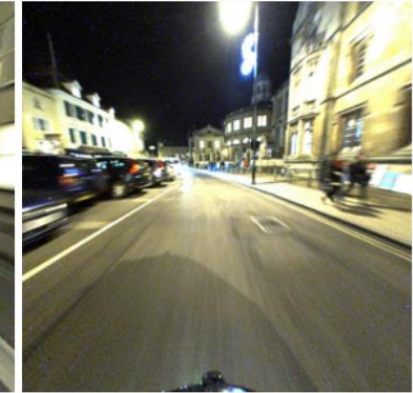
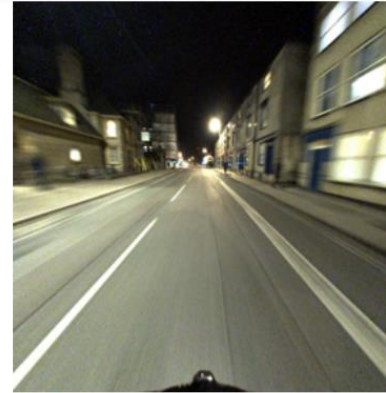
Input



Output



real nighttime



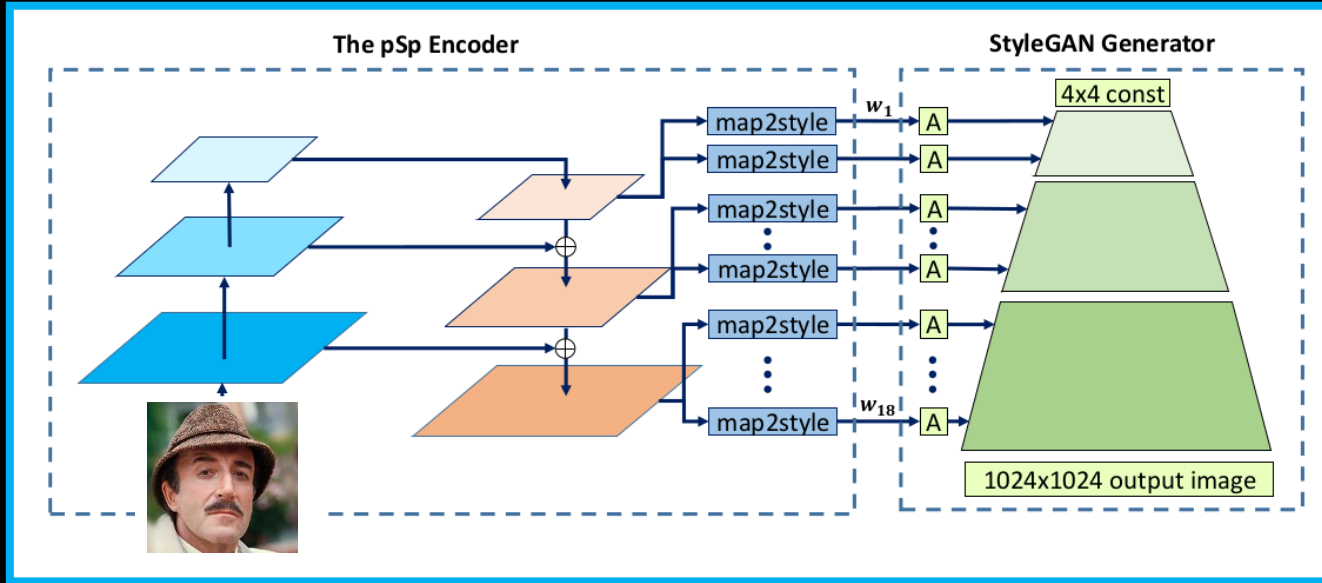
ToDayGAN  
(ours)



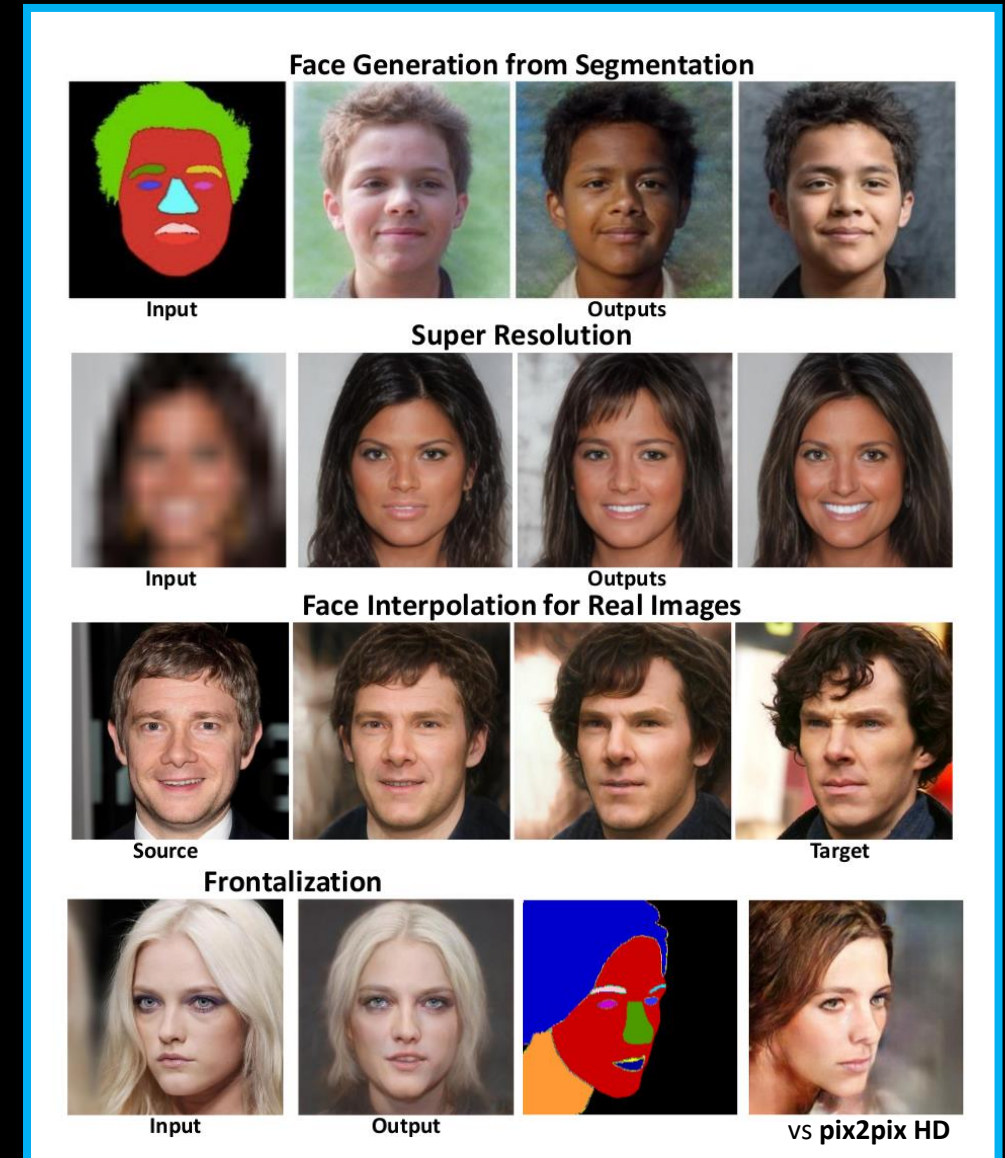
**Monet 2 Real**, and more examples  
at [junyanz.github.io/CycleGAN/](https://junyanz.github.io/CycleGAN/)

**Night 2 Day**, followed by more tasks

# Follow-up works



- **Pixel2Style2Pixel**, new research (2020) which uses a frozen pre-trained StyleGAN2 generator and **trains an encoder** for many specialized tasks while getting much higher quality than a pix2pix would →





# 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

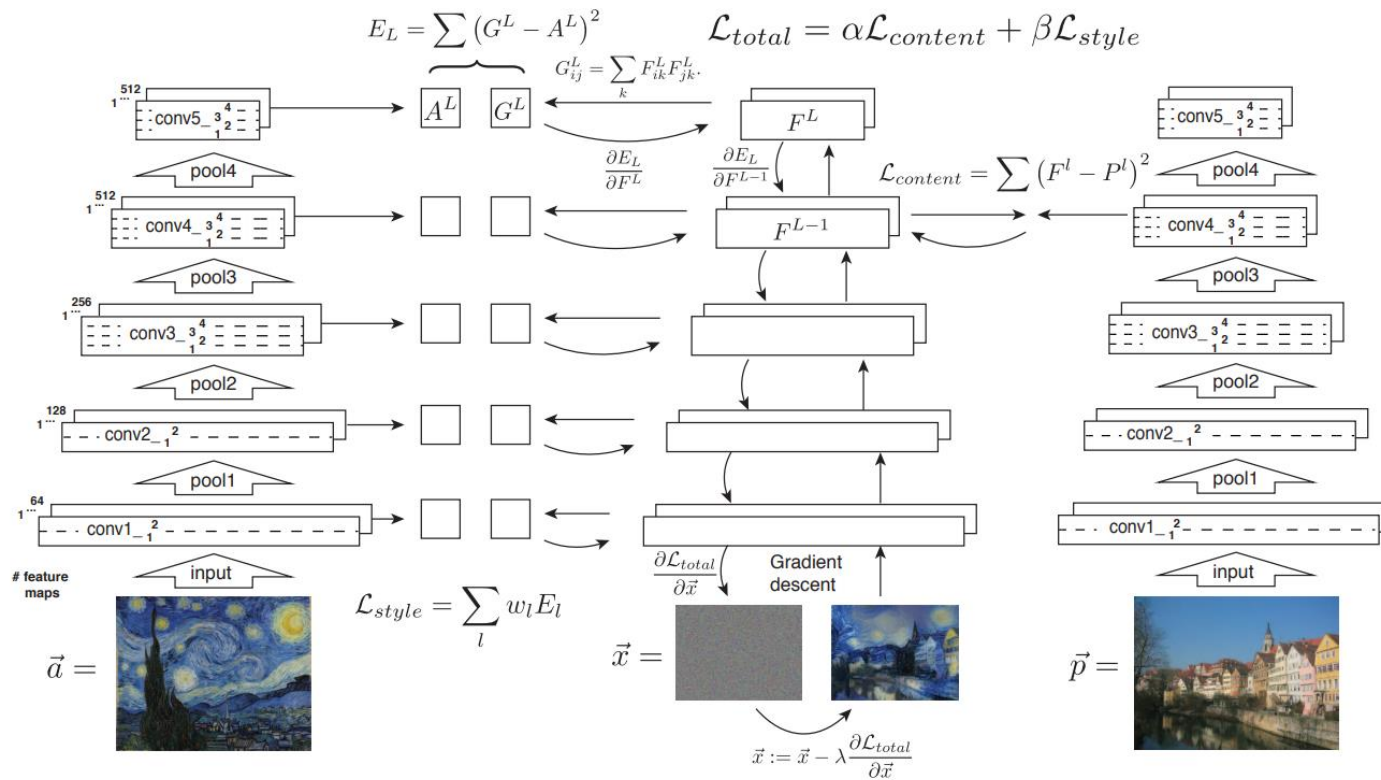
# Style Transfer

- 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

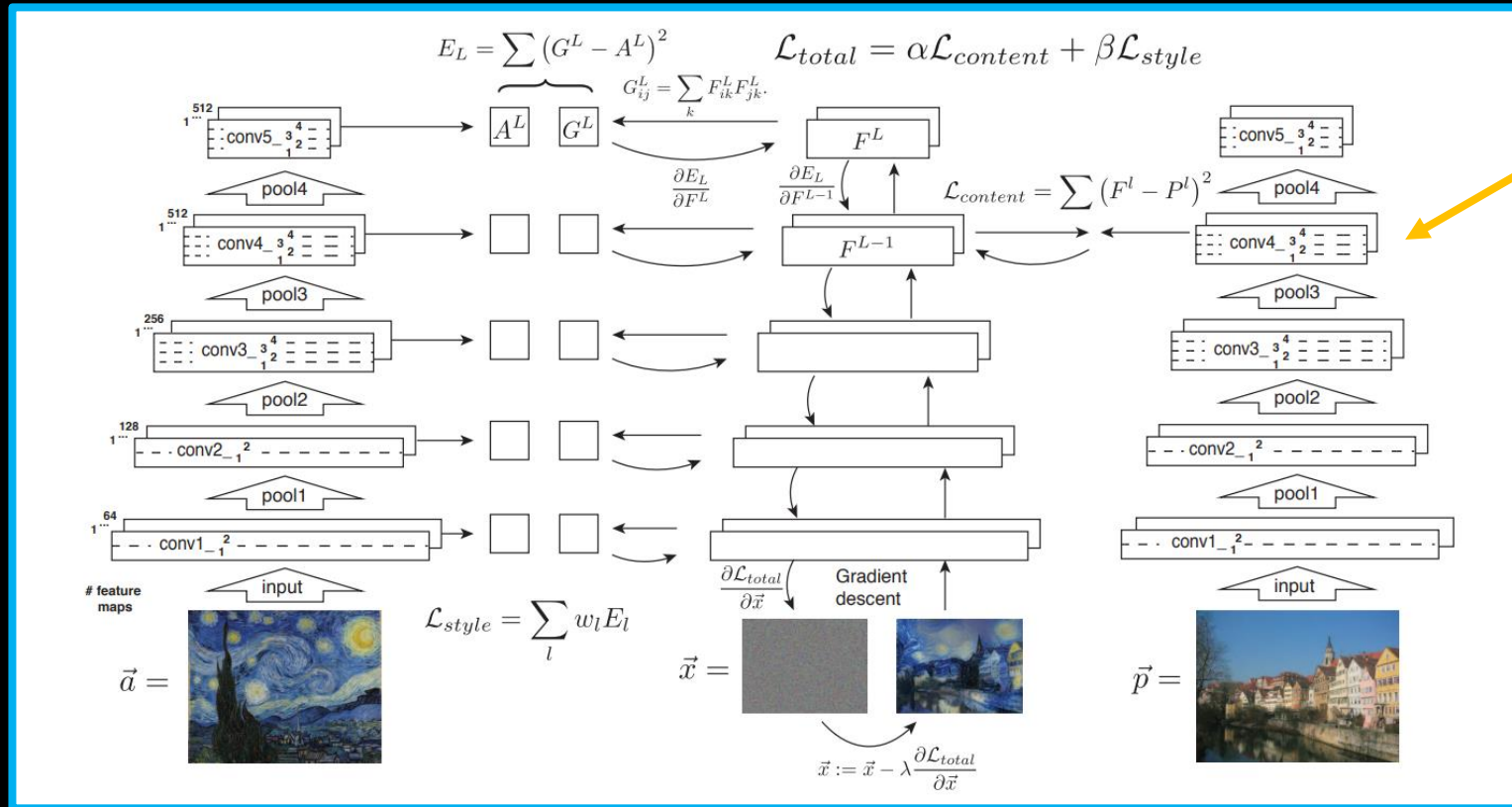
*(PS: One possible similarity if the content of low and high frequencies in music converted to spectrograms)*



# Style Transfer



# Style Transfer

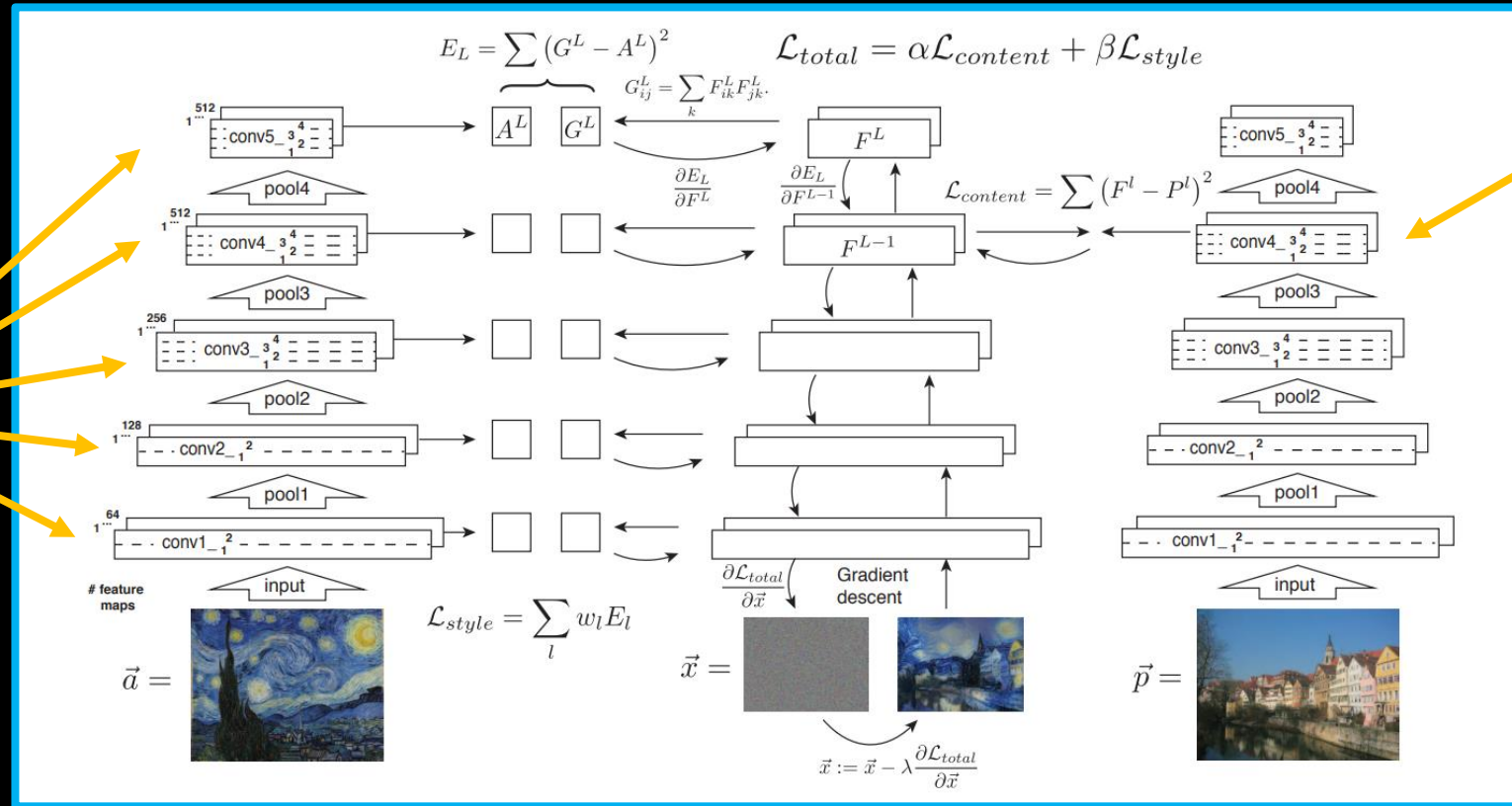


**Image content =**  
the feature  
responses in deeper  
layers of the VGG  
network



# Style Transfer

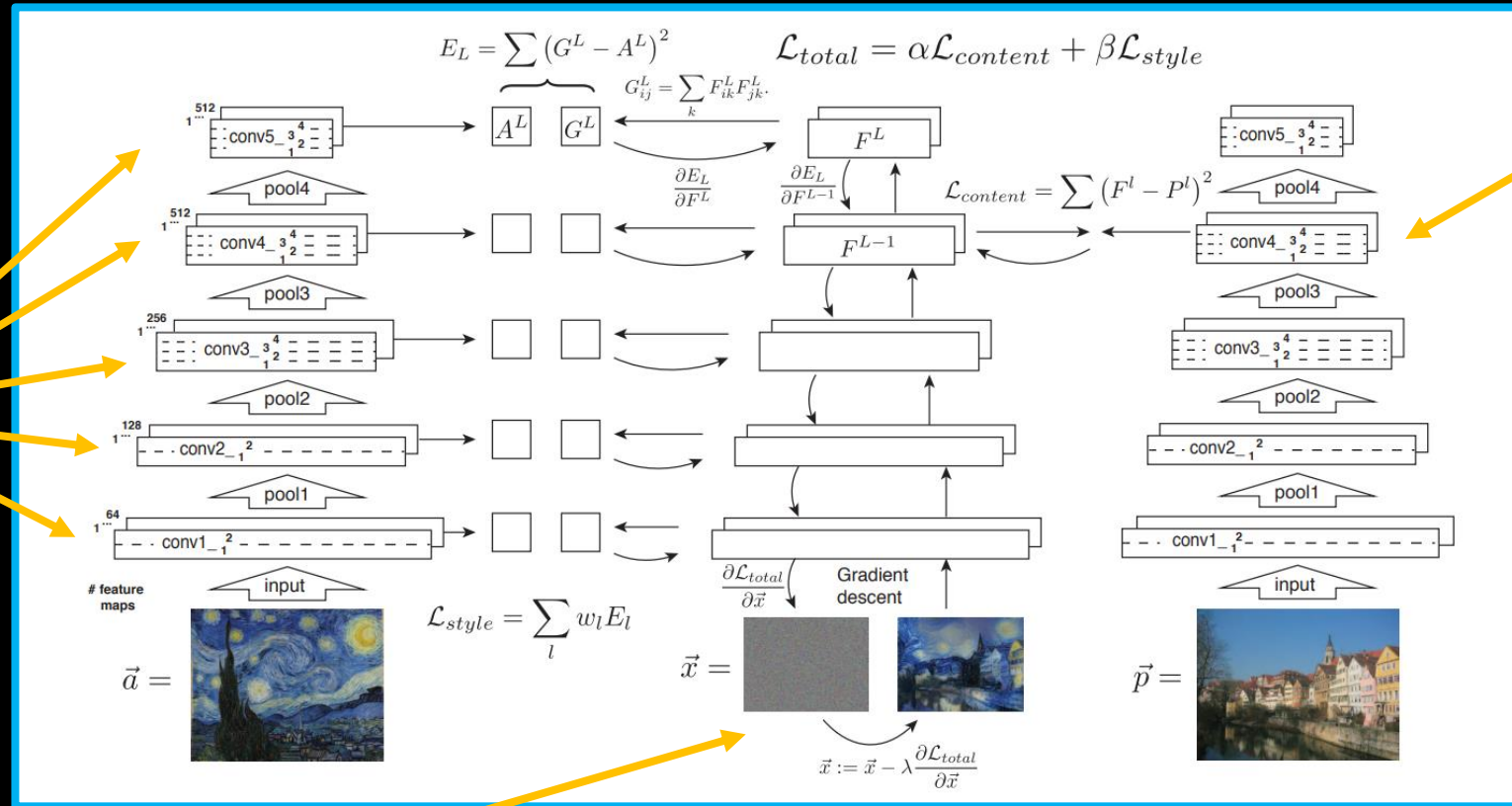
**Image style =**  
feature response  
alongside a  
selection of layers



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

# Style Transfer

**Image style =**  
feature response  
alongside a  
selection of layers



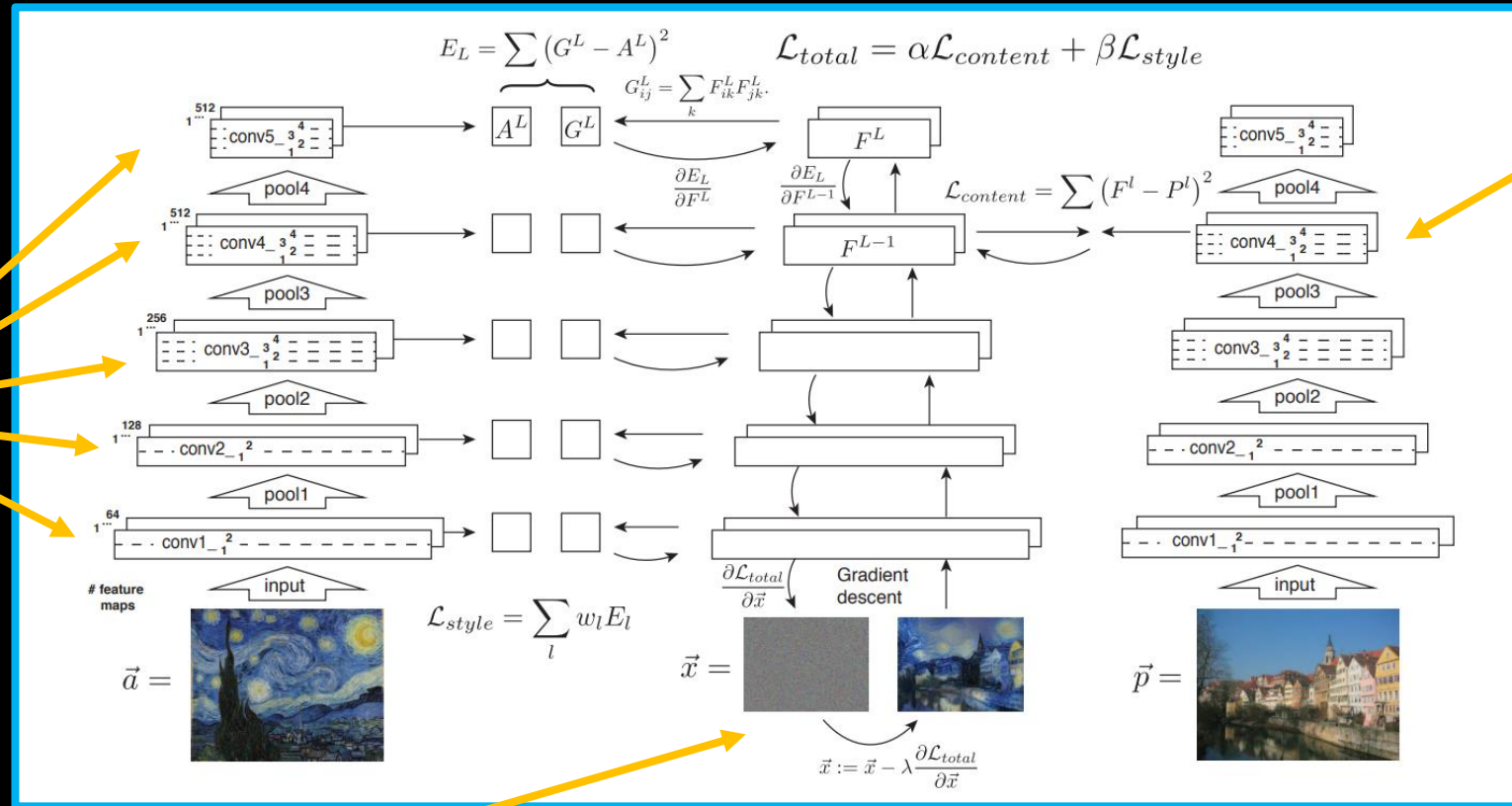
**Image content =**  
the feature  
responses in deeper  
layers of the VGG  
network

**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



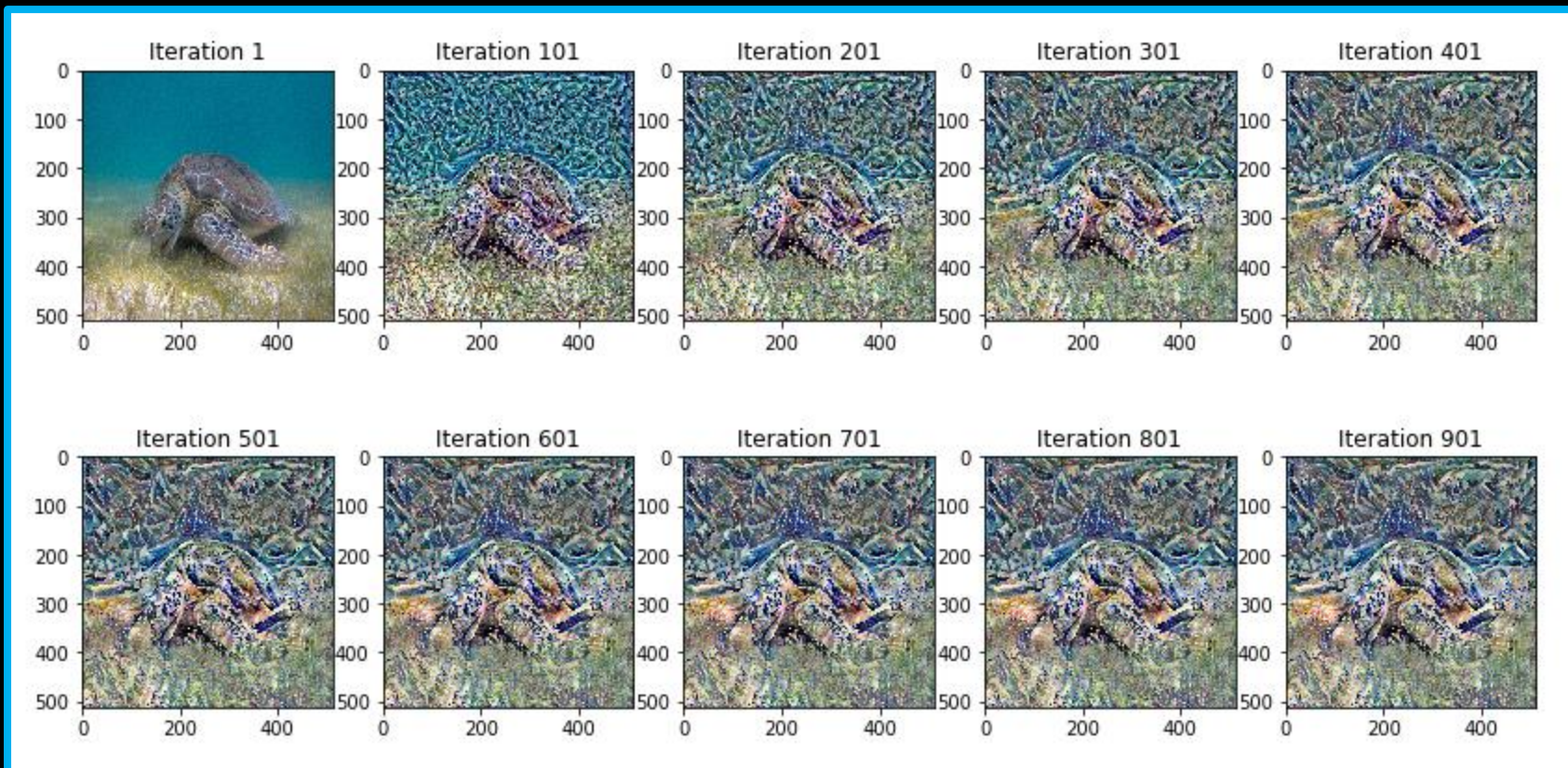
**Image content =**  
the feature  
responses in deeper  
layers of the VGG  
network

**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

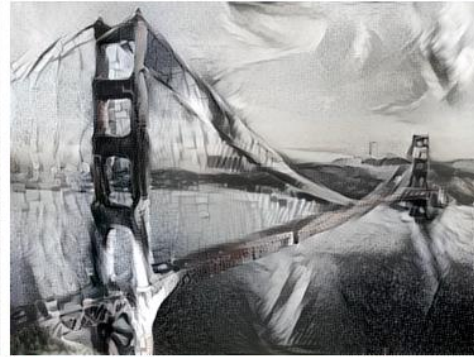
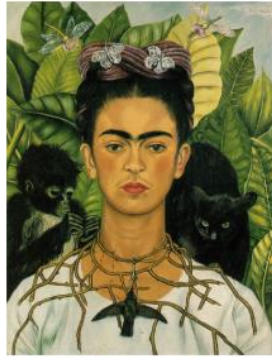
This is **relatively slow** (iterative optimization of the input image), later papers sped it up by using an **image2image translation method** with feed-forward networks.

# Style Transfer





# Style Transfer

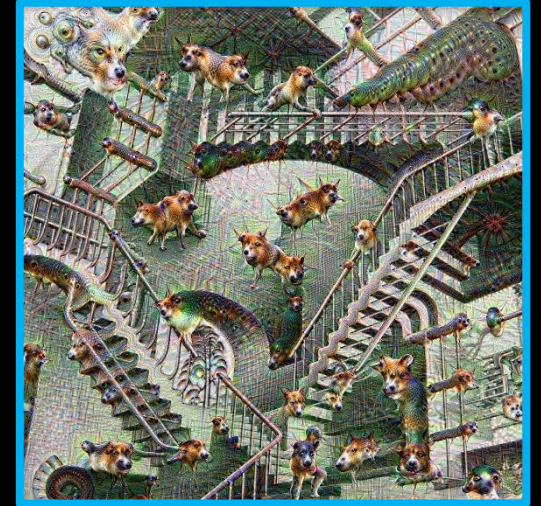


# Style Transfer

- **Online demos:** [deepart.io/latest/](https://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

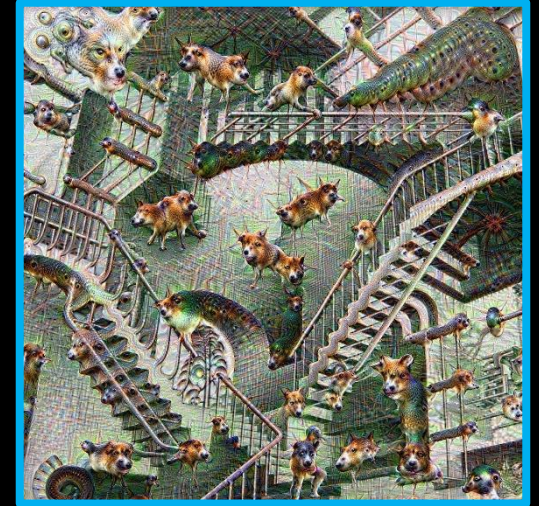
- Deep dream is a method which was originally used to **visualize what a network has learned**. It works on **image optimization** principle (as did the first version of Style Transfer).



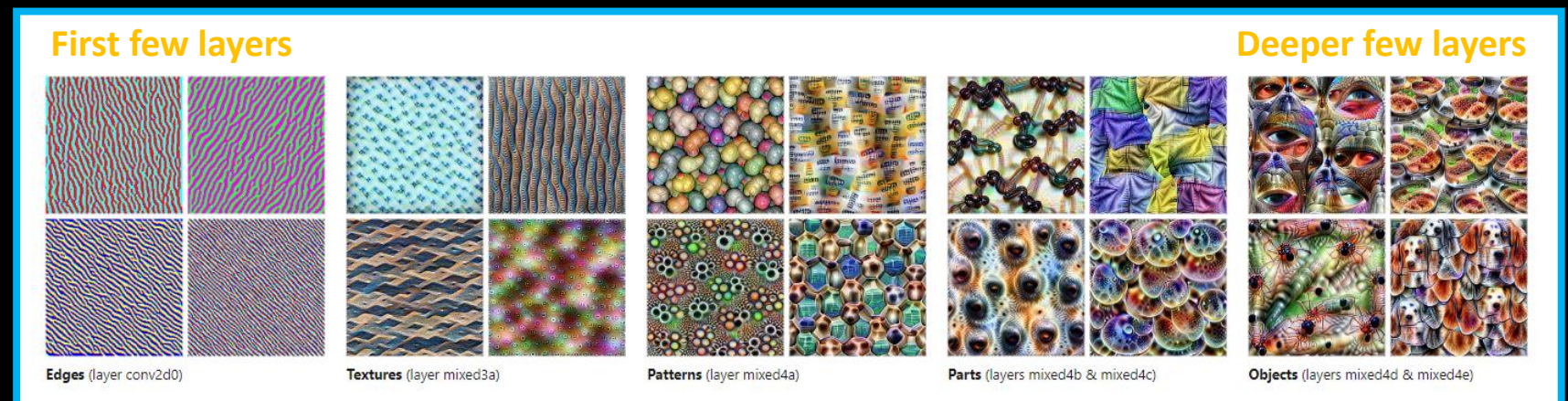


# Deep Dream

- 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:

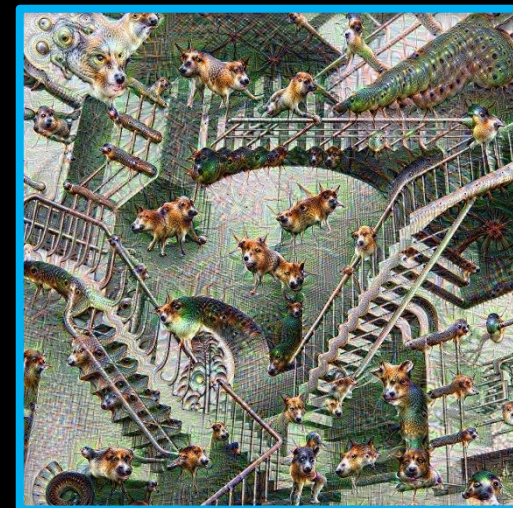


(PS: image [source](#), features [viz.](#), illustrational [video](#))

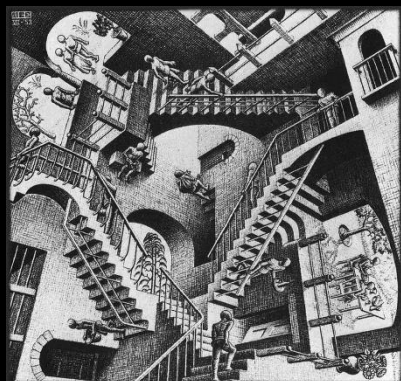


# Deep Dream

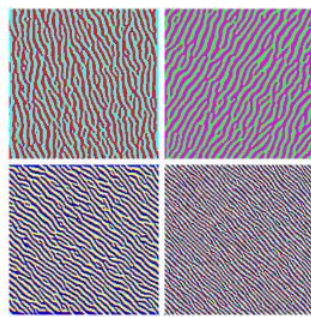
- Deep dream is a method which was originally used to **visualize what a network has learned**. It works on **image optimization** principle (as did the first version of Style Transfer).



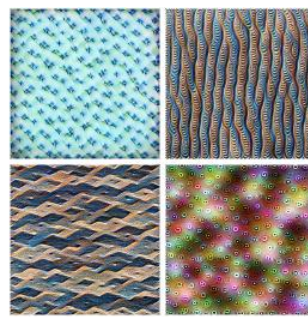
Change this **original image** so that it activates a **selected feature** with the highest possible force!



First few layers



Edges (layer conv2d0)



Textures (layer mixed3a)



Patterns (layer mixed4a)



Parts (layers mixed4b & mixed4c)

Deeper few layers



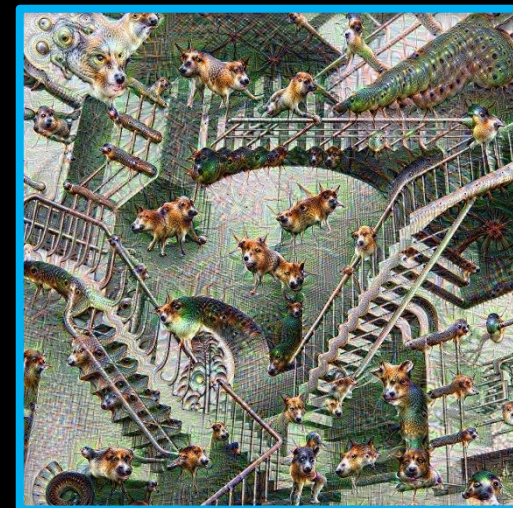
Objects (layers mixed4d & mixed4e)

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

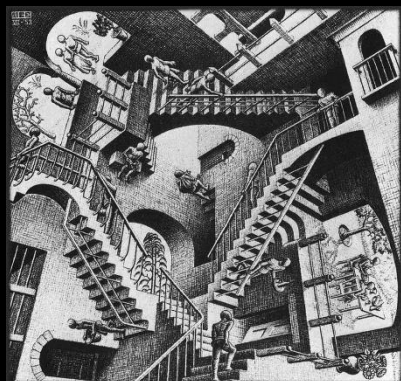


# Deep Dream

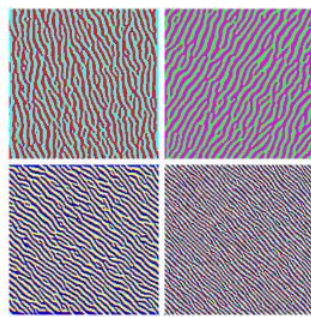
- Deep dream is a method which was originally used to **visualize what a network has learned**. It works on **image optimization** principle (as did the first version of Style Transfer).



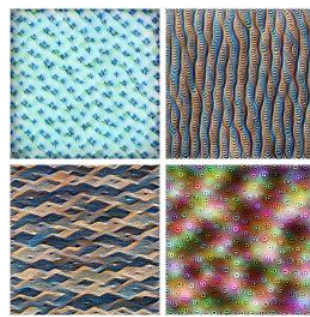
Change this **original image** so that it activates a **selected feature** with the highest possible force!



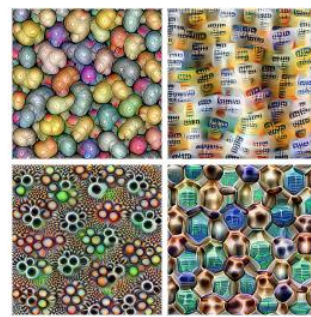
First few layers



Edges (layer conv2d0)



Textures (layer mixed3a)



Patterns (layer mixed4a)



Parts (layers mixed4b & mixed4c)

Deeper few layers



Objects (layers mixed4d & mixed4e)



**selected feature**

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

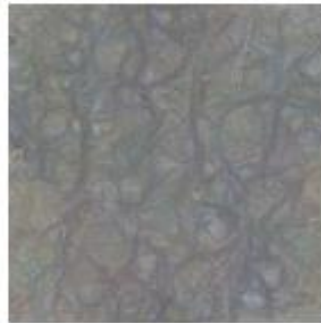
# Deep Dream

## Iterations:

Starting from random noise, we optimize an image to activate a particular neuron (layer mixed4a, unit 11).



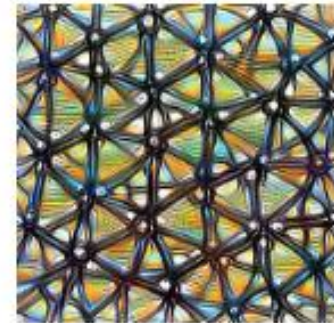
Step 0



Step 4



Step 48



Step 2048

# Deep Dream

## As a network visualization technique ...

Deep dreaming can reveal information stored inside the network ...  
Recall shapes and content information from the originally used dataset.

- Model released by Yahoo to identify NSFW content (without releasing the sensitive information about the dataset): [github.com/yahoo/open\\_nsfw](https://github.com/yahoo/open_nsfw)
- Which was (*of course* ...) soon followed by using a deep-dream–like technique: [open\\_nsfw.gitlab.io/](https://open_nsfw.gitlab.io/)
  - Interesting usage of the network to generate *suggestive imagery* (changing shapes of landscapes imagery, etc.)



# Deep Dream

- **Online demos:** [dreamscopeapp.com](https://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/](https://distill.pub/2017/feature-visualization/)



# Links and additional readings:

- **Online pix2pix demos:** [affinelayer.com/pixsrv/](https://affinelayer.com/pixsrv/)
  - **Follow-up papers:** [pix2pixHD](#) (with high. res.), [vid2vid](#) (with frame to frame consistency)
- **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](#)

# End of the lecture

\*) PS: follows material for the practical session ...

# AI for the Media

## Week 9, Domain 2 Domain



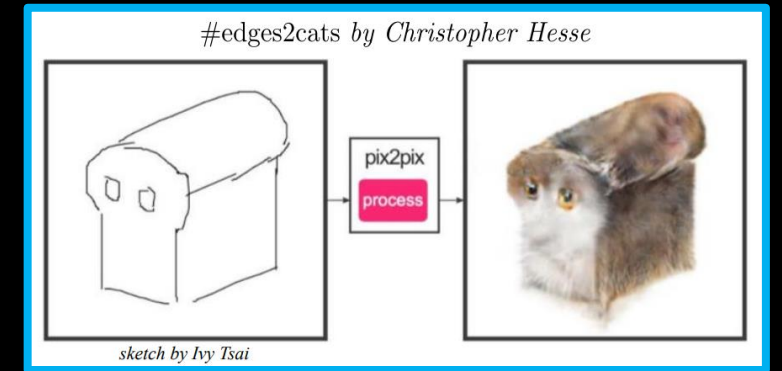
Practical: pix2pix training



# Practical: Domain 2 Domain

## Training Pix2Pix models

Namely using the sketch2image with our own data.

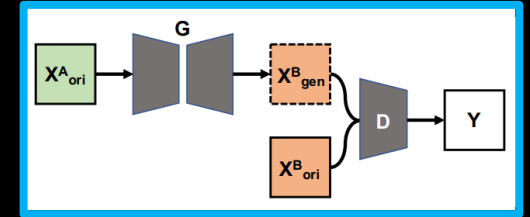


Continue with code on Github:

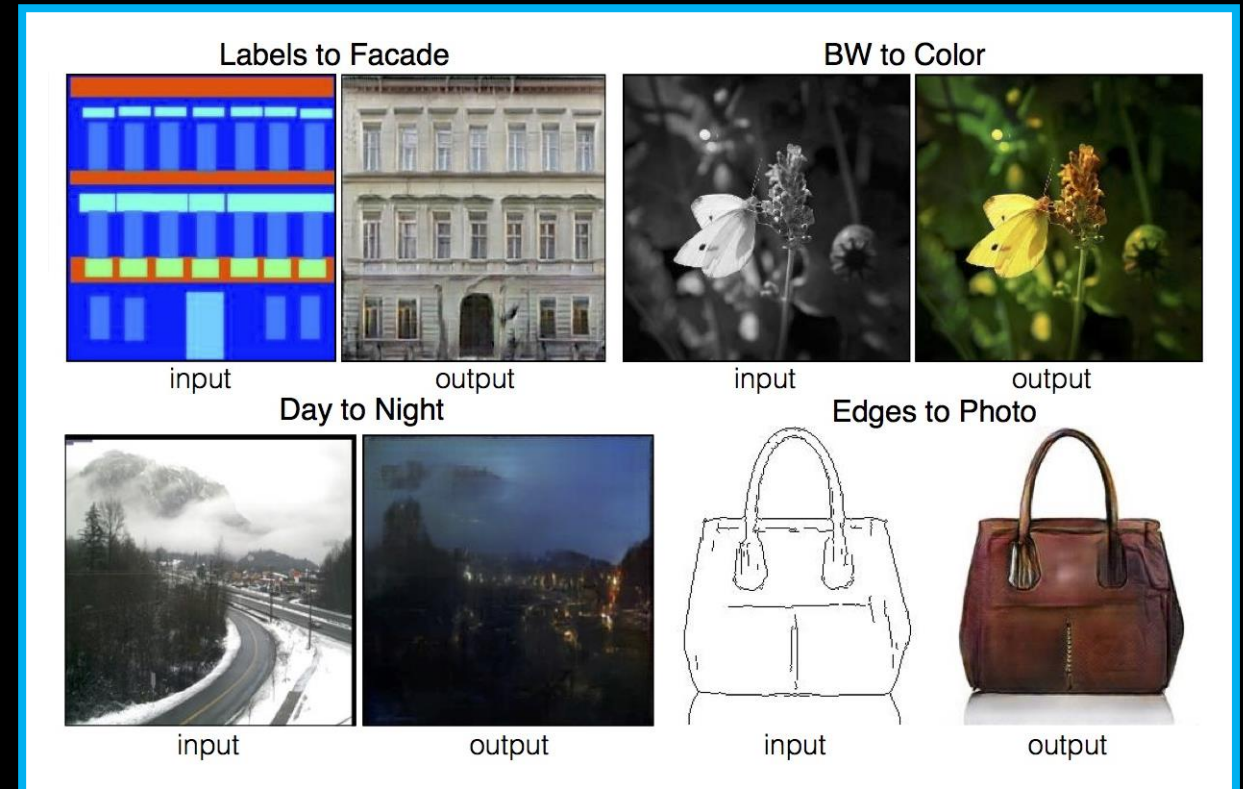
- Repo: [github.com/previtus/ci](https://github.com/previtus/ci) AI for the Media
- **Notebook** directly:  
[week09 domain-to-domain/aim09 pix2pix keras.ipynb](#)



# Pix2Pix



- You might recall that we will need **paired data** in our dataset:
  - Data from domain **A**
  - Data from domain **B**
- And we will be learning the **translation between A to B**



The end