Computer vision tasks

Classification

Multi-label classification

y = 0 1 0

y = 1 1 0

y = 1 0 1

Activation function → Sigmoid

Binary cross-entropy loss

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} y_{ij} \ln p(y_{ij} \mid x_i) + (1 - y_{ij}) \ln(1 - p(y_{ij} \mid x_i))$$

Multi-class classification

y = 1 0 0

y = 0 1 0

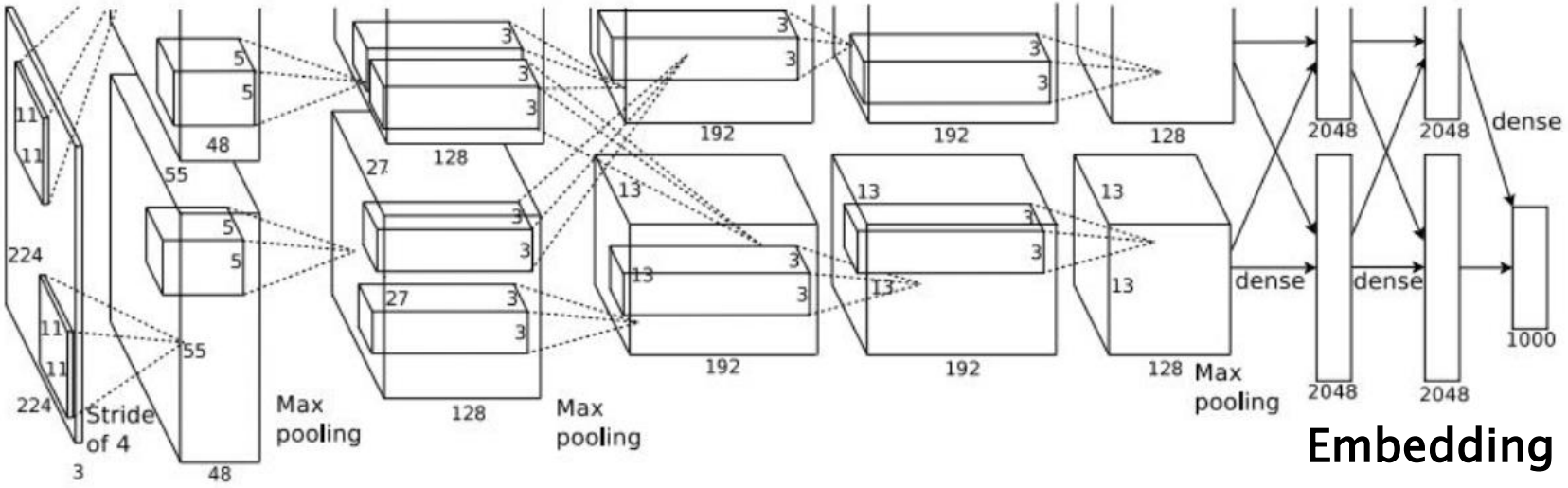
y = 0 0 1

Activation function → Softmax

Cross-Entropy Loss Function

$$L = -\frac{1}{N}\sum_{i} \ln p(c = y_i \mid x_i)$$

Classification



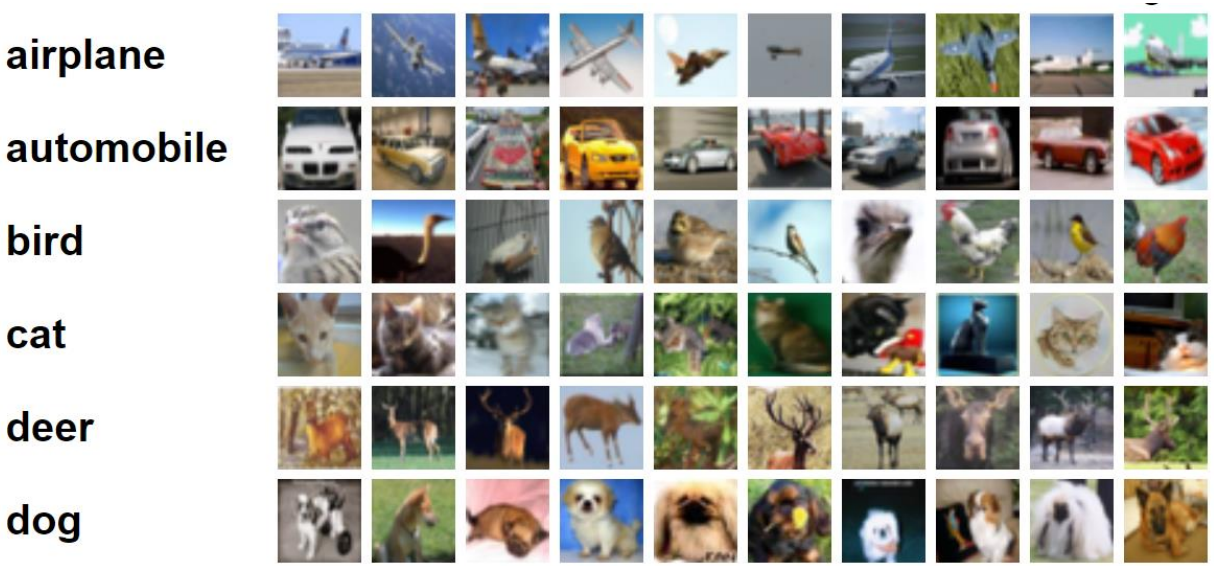
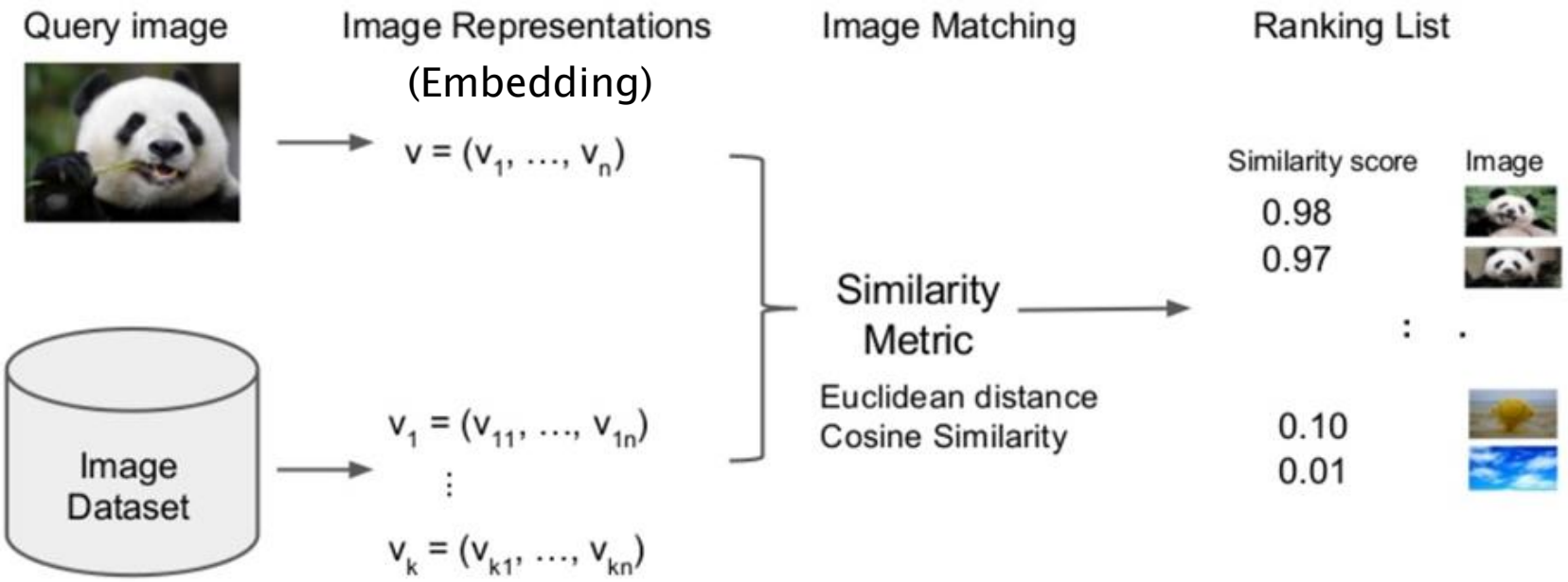
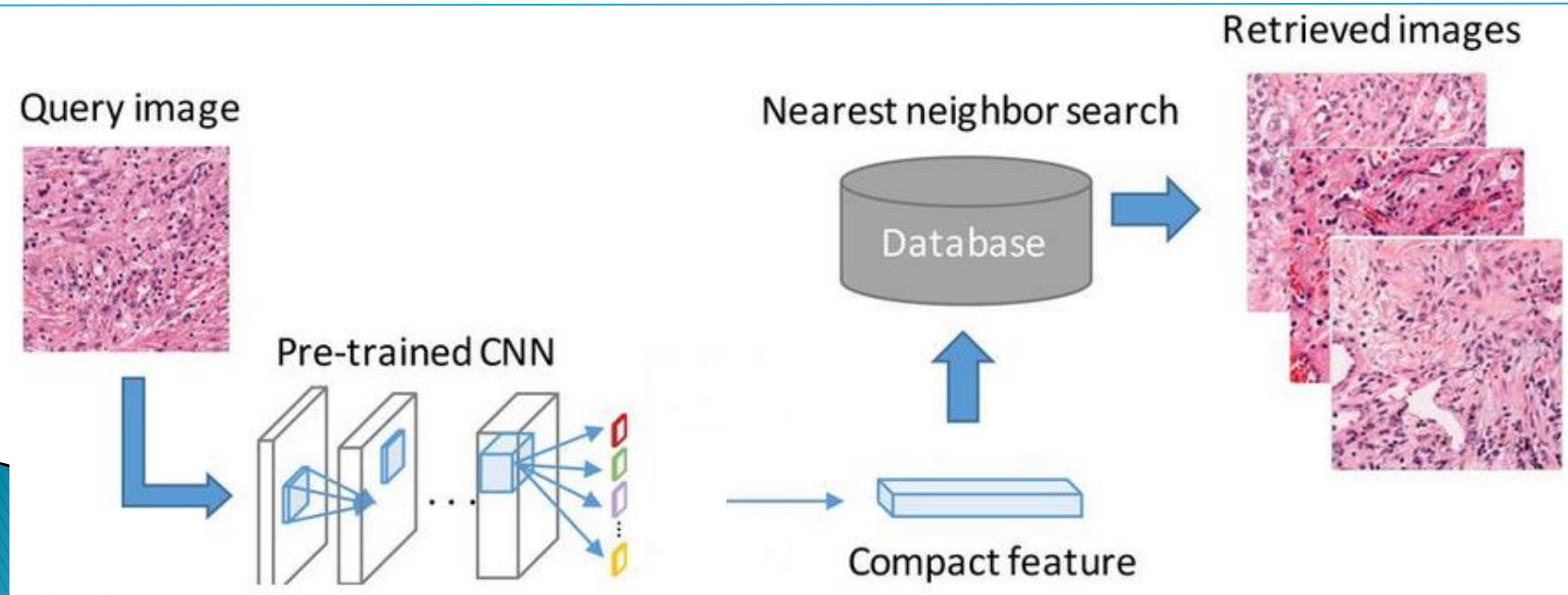


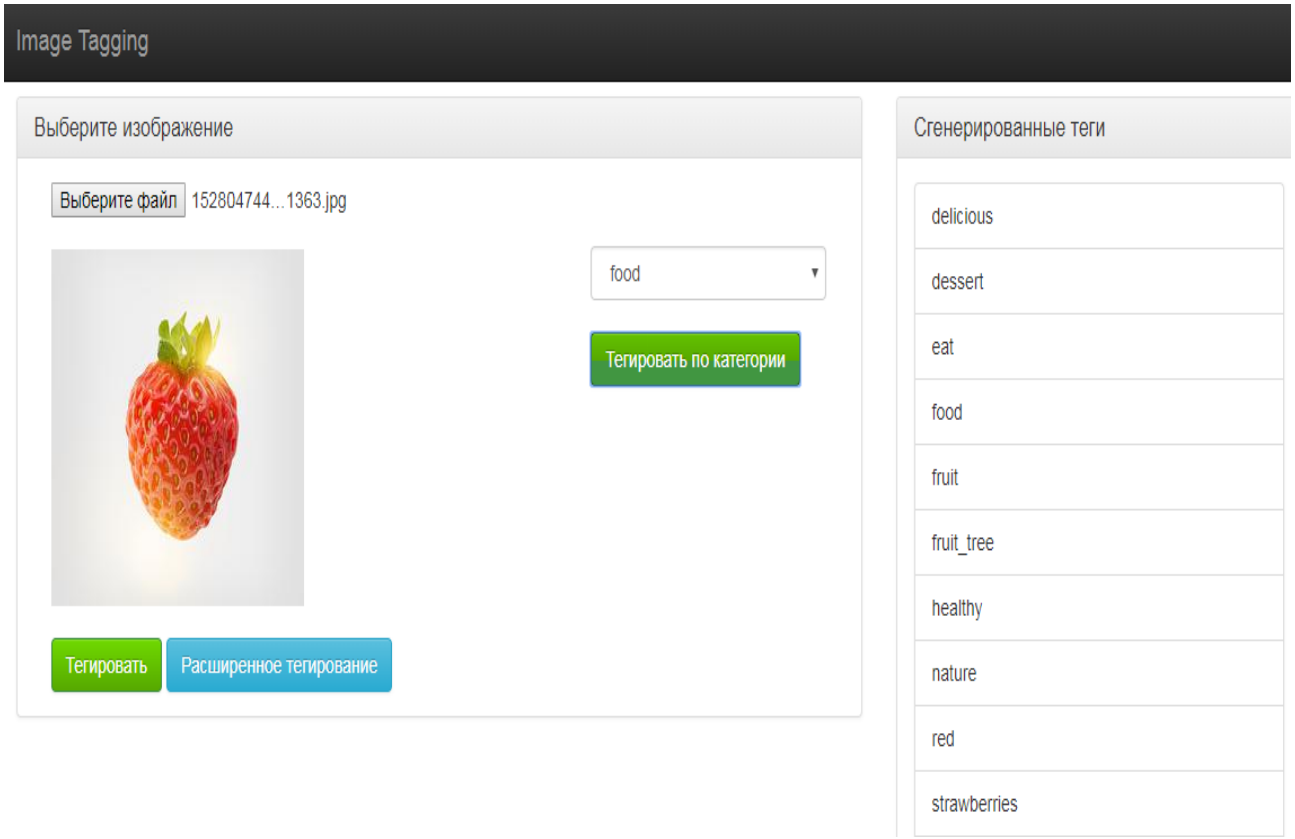
Image retrieval





https://www.slideshare…vigiro/deep-learning-for-computer-vision-image-retrieval-upc-2016

Image tagging



When classification doesn't work

- extremely unbalanced dataset with a huge number of classes
- objects of new classes (not from the training set)

Megaface

17 categories of clothing (jackets, jeans, shorts, etc.)
~8 thousand classes (articles of specific products)
The median class size is 5 images.

Distractors

1 Million Photos

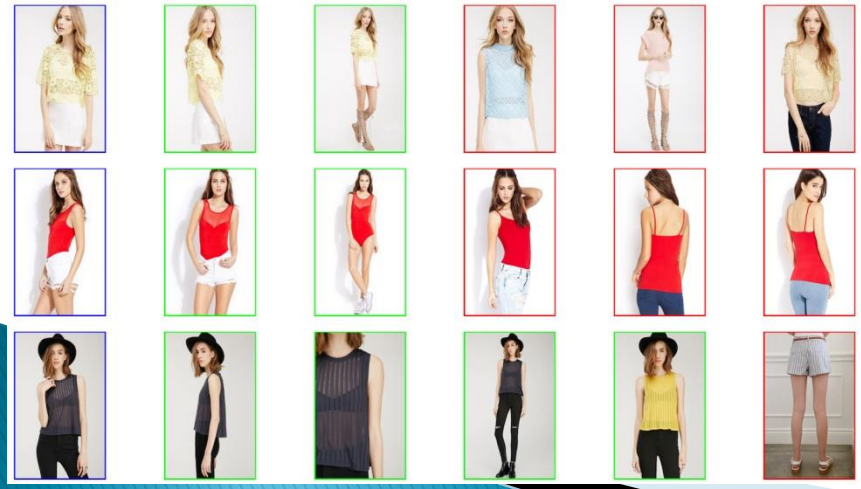
690,572 Unique Users

Training Set

4.7 Million Photos

672,057 Unique Identities

7 Mean photos / person (3 min, 2469 max)



DeepFashion

Metric Learning

Triplet Loss

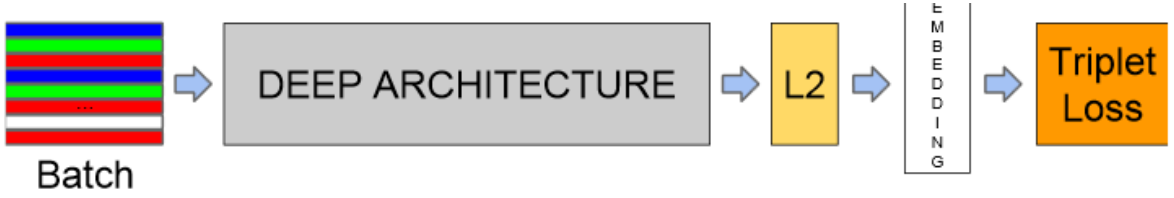


Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

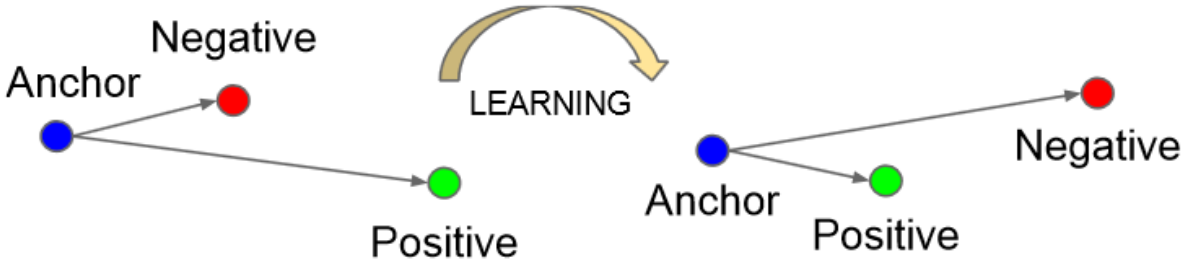


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

$$L = \sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

https://arxiv.org/pdf/1503.03832.pdf

TensorFlow Addons Losses: TripletSemiHardLoss

Run in Google Colab | View source on GitHub | Download notebook

Overview

This notebook will demonstrate how to use the TripletSemiHardLoss function in TensorFlow Addons.

Resources:

- FaceNet: A Unified Embedding for Face Recognition and Clustering
- Oliver Moindrot's blog does an excellent job of describing the algorithm in detail

TripletLoss

As first introduced in the FaceNet paper, TripletLoss is a loss function that trains a neural network to closely embed features of the same class while maximizing the distance between embeddings of different classes. To do this an anchor is chosen along with one negative and one positive sample.

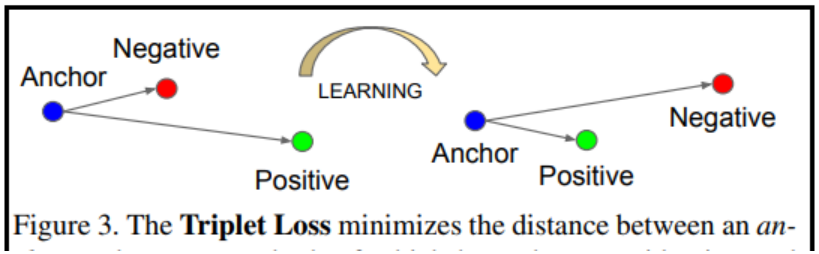


Figure 3. The **Triplet Loss** minimizes the distance between an *an-*

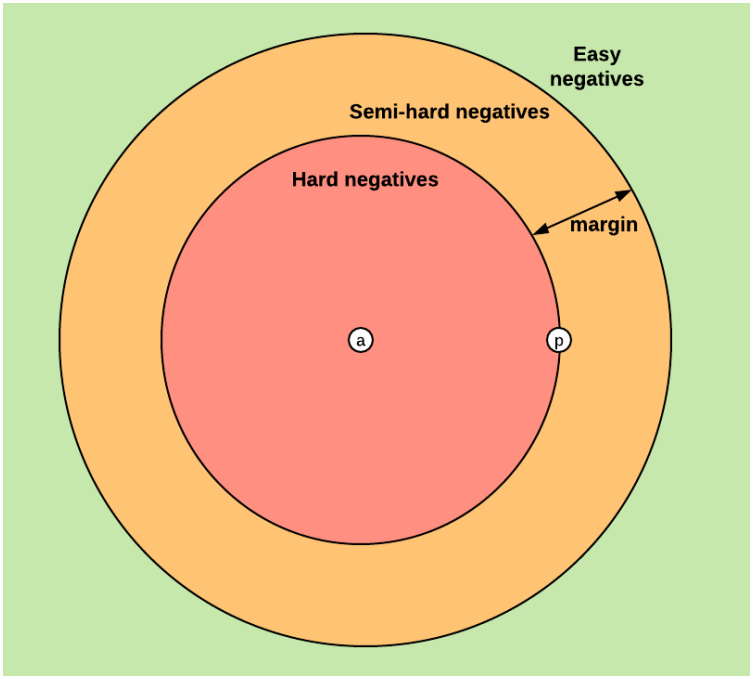
https://www.tensorflow.org/addons/tutorials/losses_triplet

Triplet mining

Based on the definition of the loss, there are three categories of triplets:

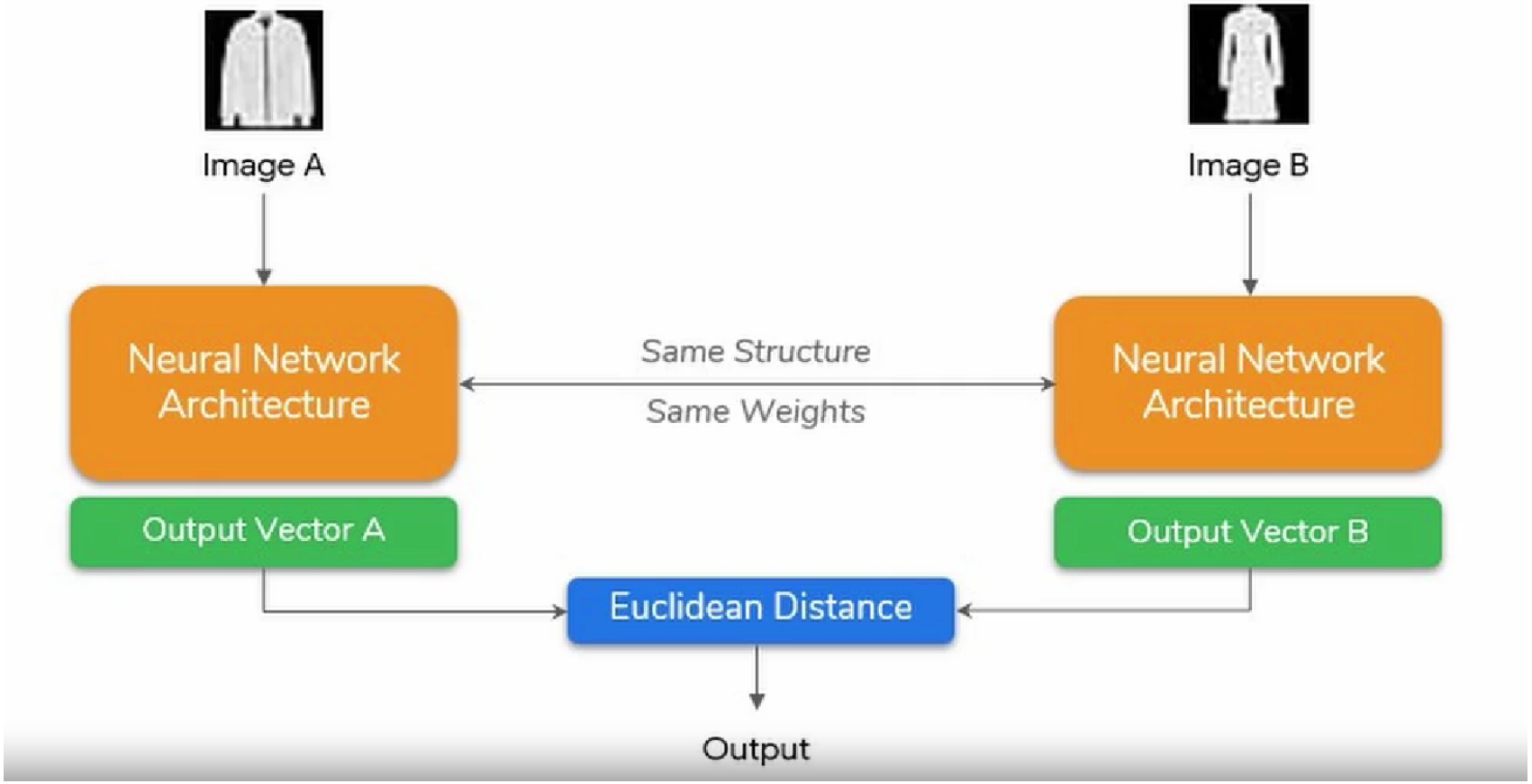
- **easy triplets**: triplets which have a loss of 0, because $d(a,p) + margin < d(a,n)$
- **hard triplets**: triplets where the negative is closer to the anchor than the positive, i.e. $d(a,n) < d(a,p)$
- **semi-hard triplets**: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss: $d(a,p) < d(a,n) < d(a,p) + margin$

Each of these definitions depend on where the negative is, relatively to the anchor and positive. We can therefore extend these three categories to the negatives: **hard negatives**, **semi-hard negatives** or **easy negatives**.



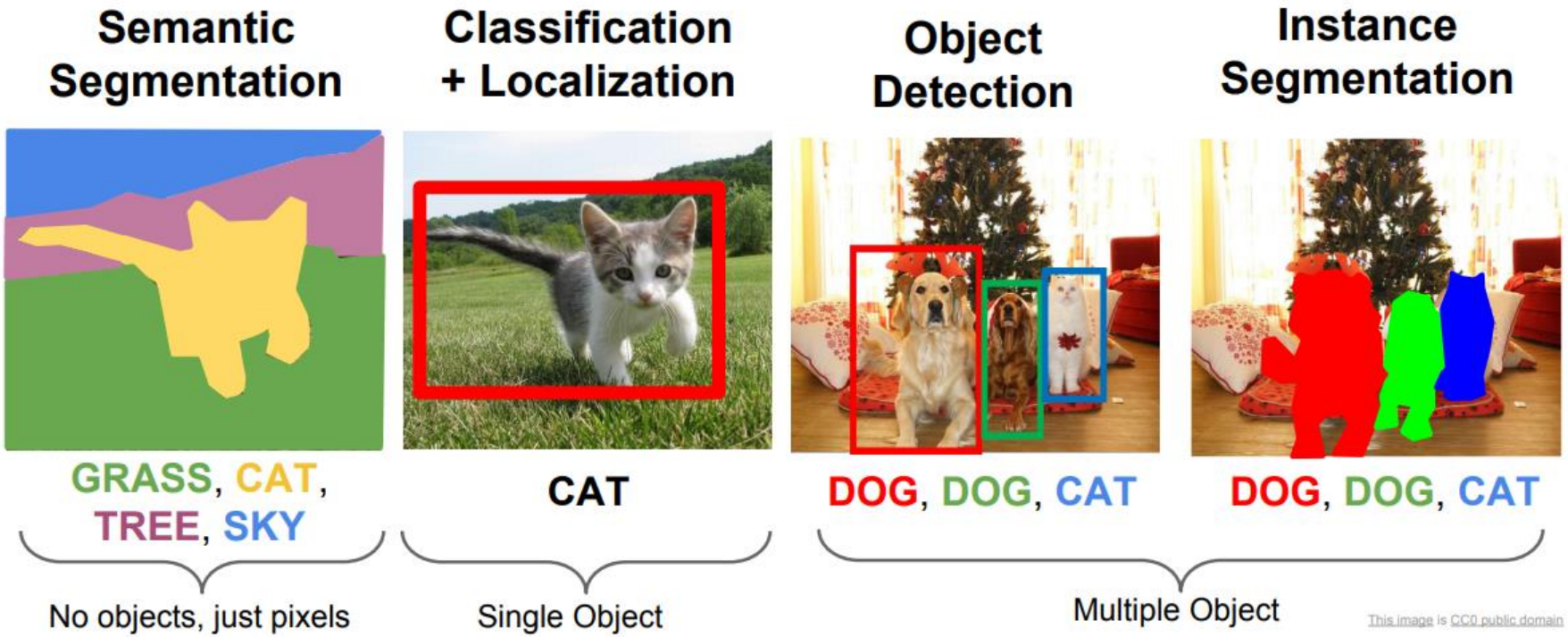
https://omoindrot.github.io/triplet-loss

A Siamese network's architecture

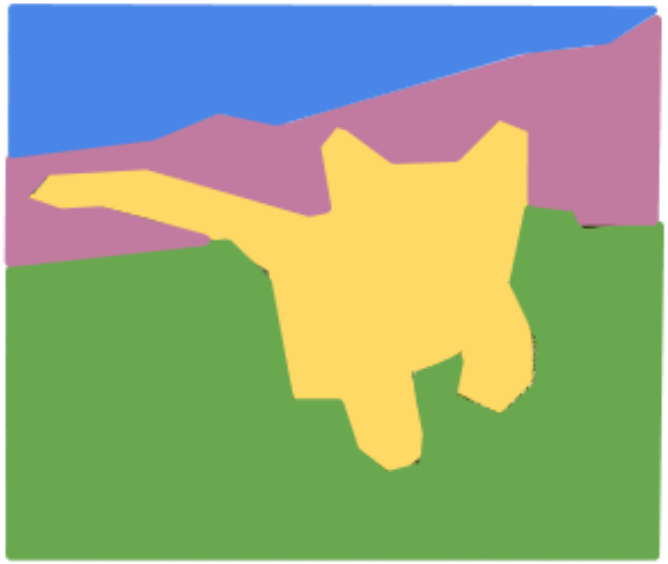


https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/lecture/ynUV8/siamese-network-a-multiple-input-model

Semantic segmentation, Object localization, Object detection

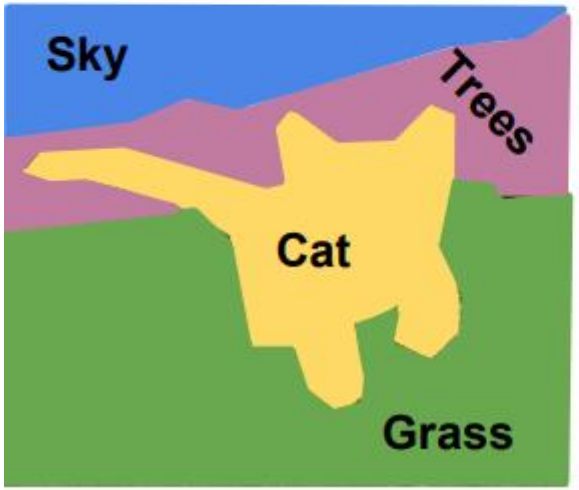


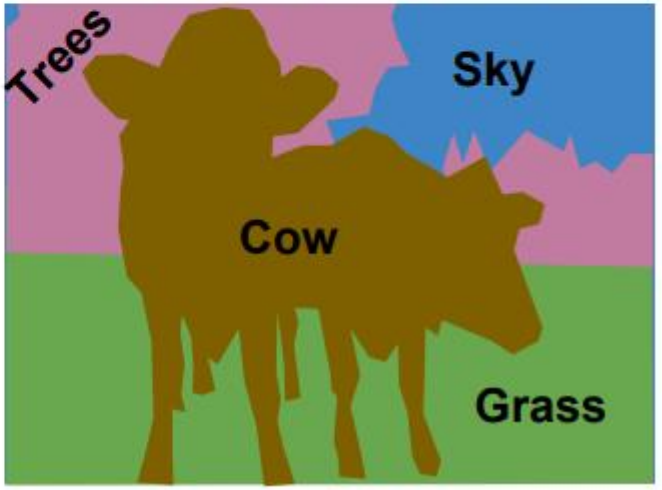
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf



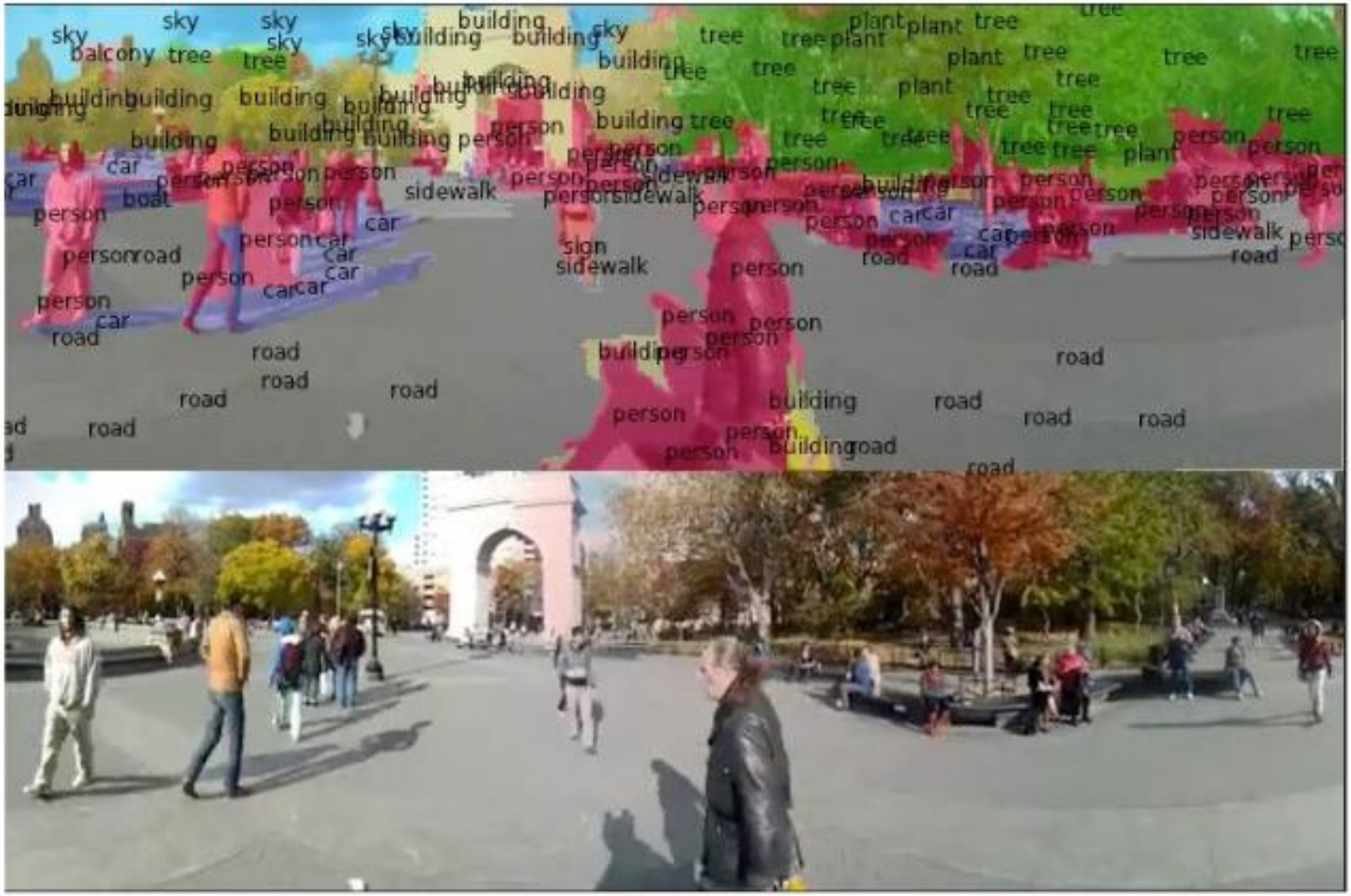
GRASS, CAT,
TREE, SKY

Semantic Segmentation

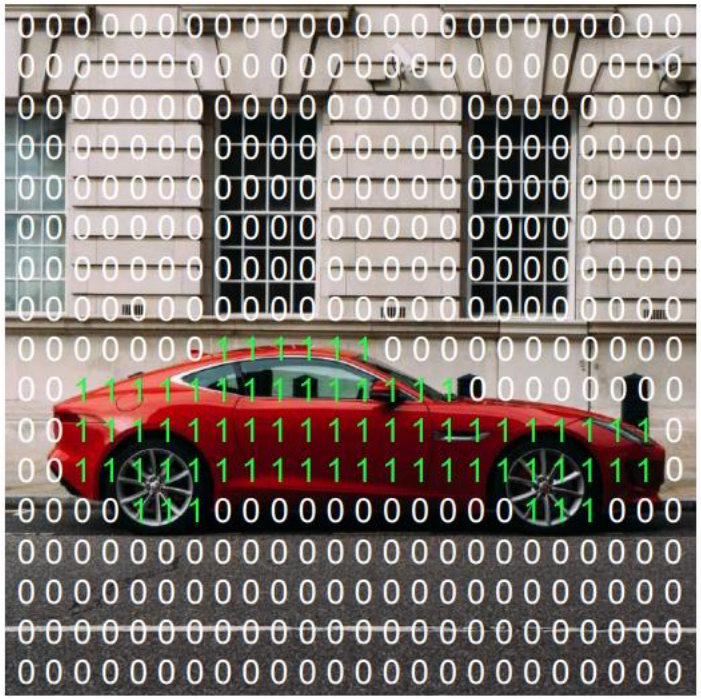




Segmentation



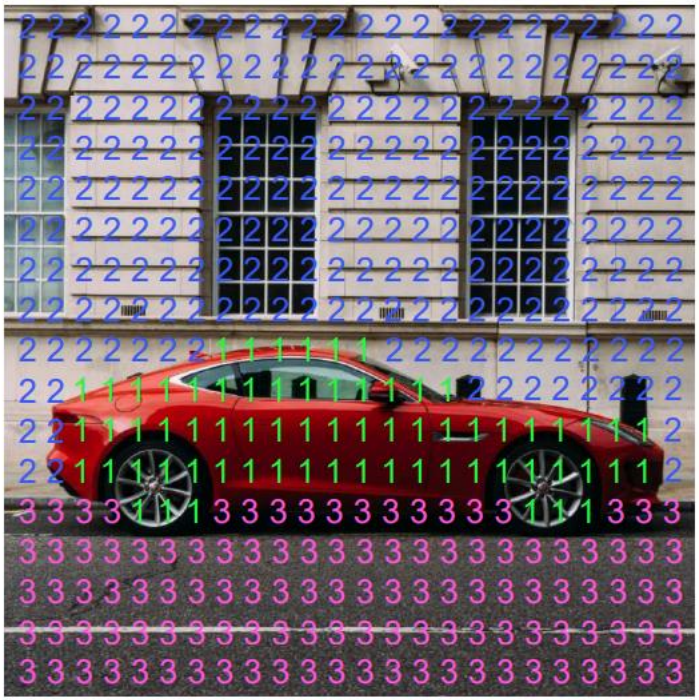
Per-pixel class labels



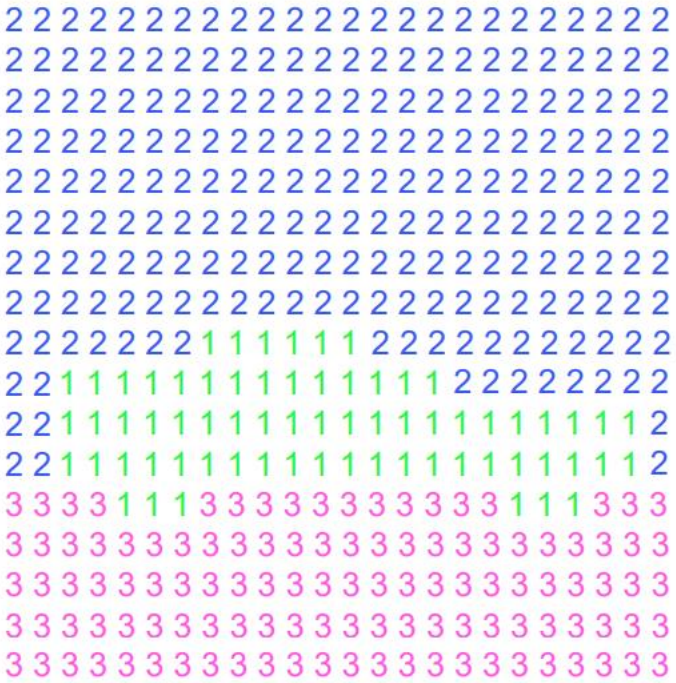
1. Car
0. Not Car

https://www.coursera.org/learn/convolutional-neural-networks/home/info

Per-pixel class labels



1. Car
2. Building
3. Road



Segmentation Map

https://www.coursera.org/learn/convolutional-neural-networks/home/info

Image Segmentation Basic Architecture

224 x 224 x 3 (RGB image)

Encoder
(Feature Extractor, Down samples Image)

Downsampled Feature Map

Decoder
(Up samples feature map, generates pixel-wise label map

224 x 224 x n (Pixel map)
n = number of classes

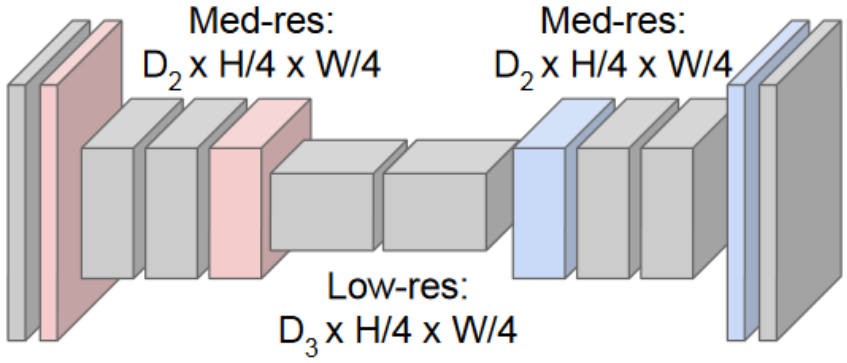
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

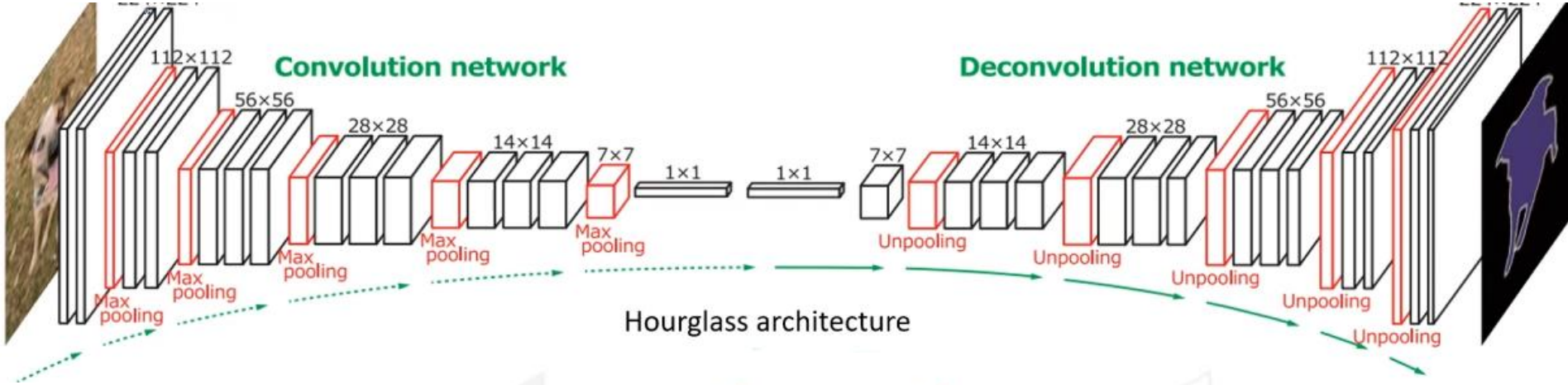
High-res:
D_1 x H/2 x W/2

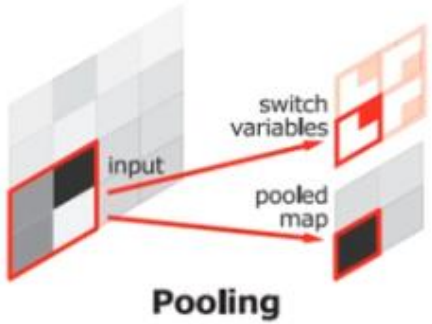
Predictions:
H x W

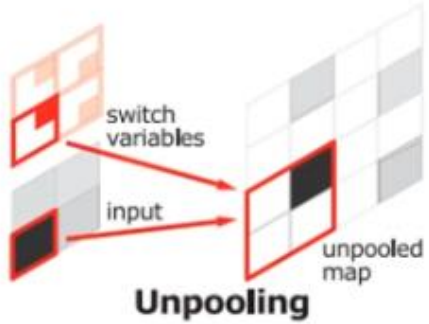
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

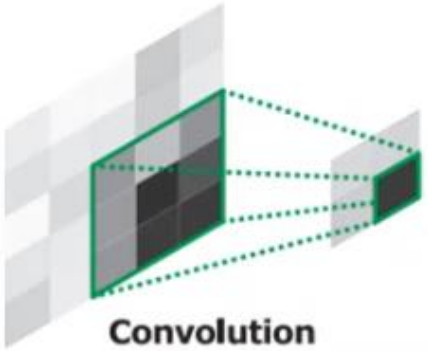
https://www.youtube.com/[redacted]v=nDPWywWRIRo&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=11

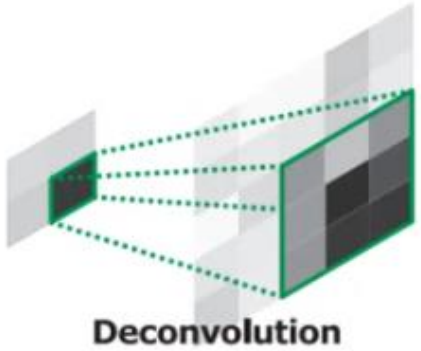
SegNet





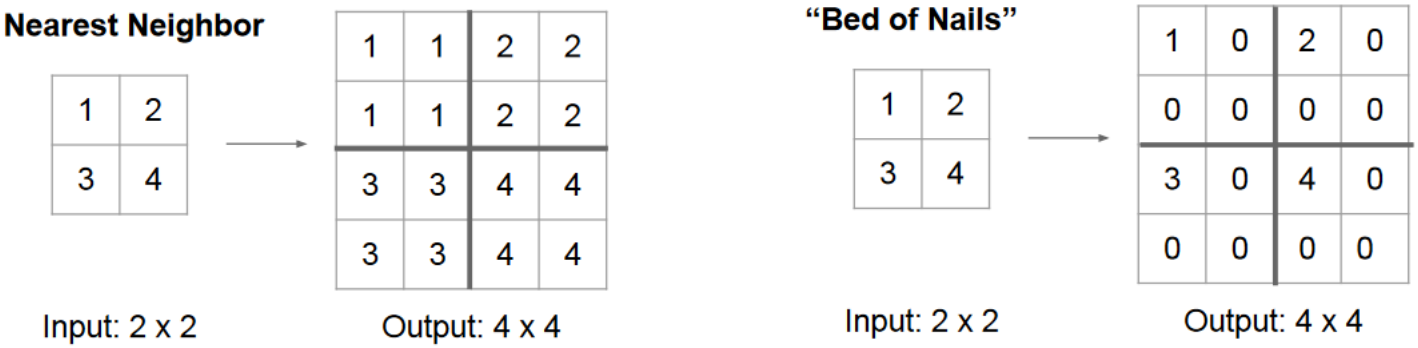




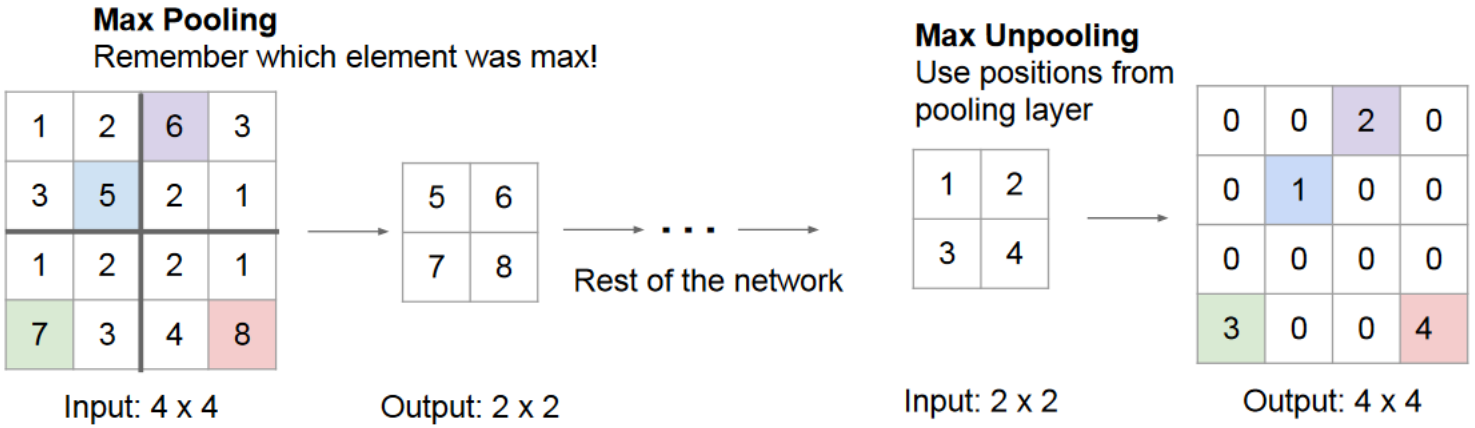


https://arxiv.org/pdf/1505.04366.pdf

In-Network upsampling: “Unpooling”



In-Network upsampling: “Max Unpooling”

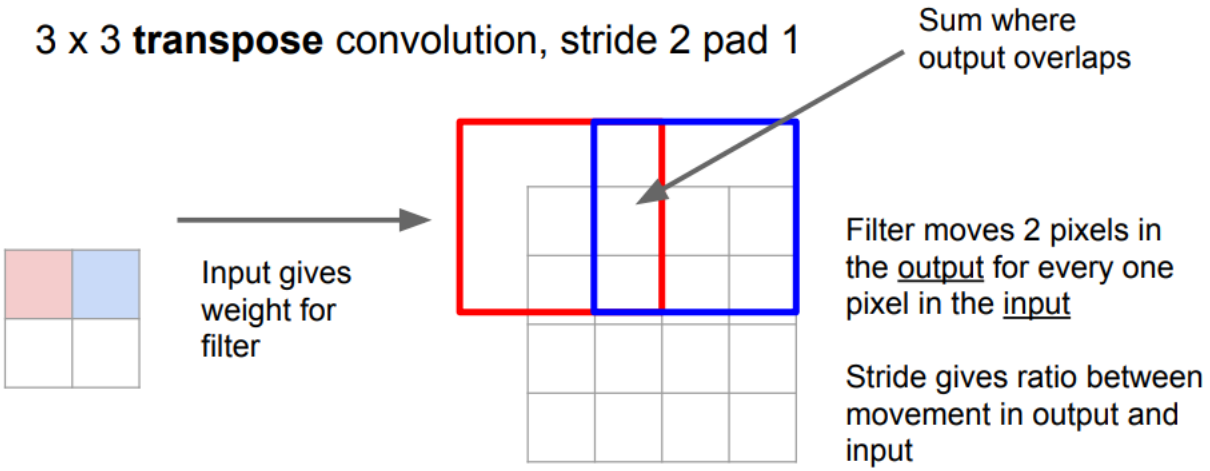


https://www.youtube.com/[illegible]v=nDPWywWRIRo&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=11

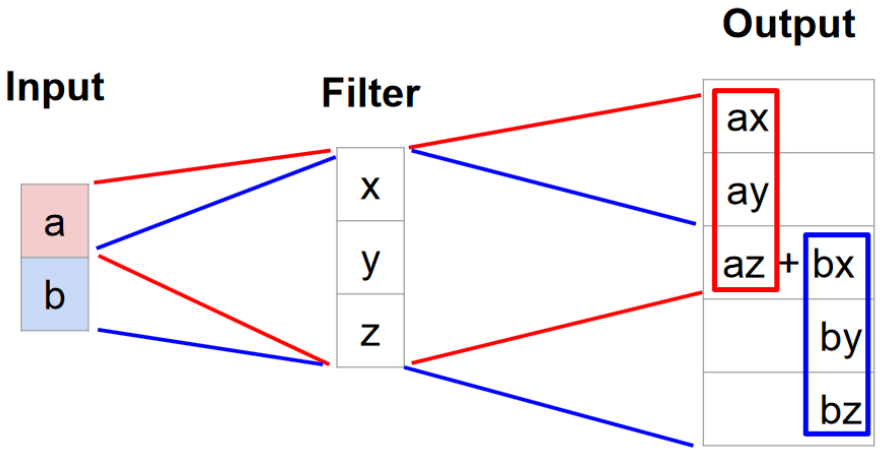
Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

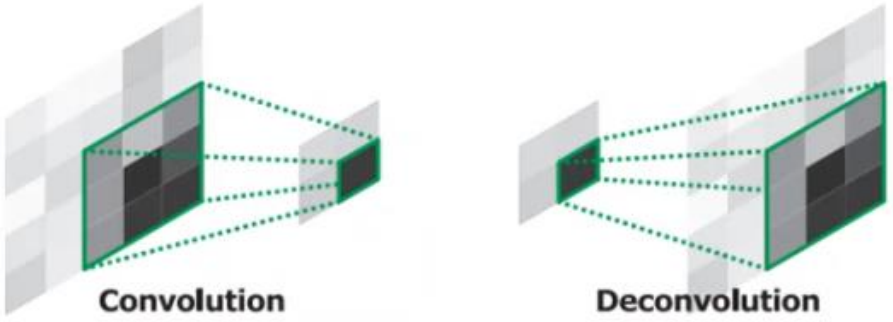
3 x 3 **transpose** convolution, stride 2 pad 1



Transpose Convolution: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

Filter

- BatchNormalization
- Bidirectional
- CategoryEncoding
- CenterCrop
- Concatenate
- Conv1D
- Conv1DTranspose
- Conv2D
- Conv2DTranspose
- Conv3D
- Conv3DTranspose
- ConvLSTM1D
- ConvLSTM2D
- ConvLSTM3D
- Cropping1D
- Cropping2D
- Cropping3D
- Dense
- DenseFeatures
- DepthwiseConv1D
- DepthwiseConv2D
- Discretization
- Dot
- Dropout

tf.keras.layers.Conv2DTranspose

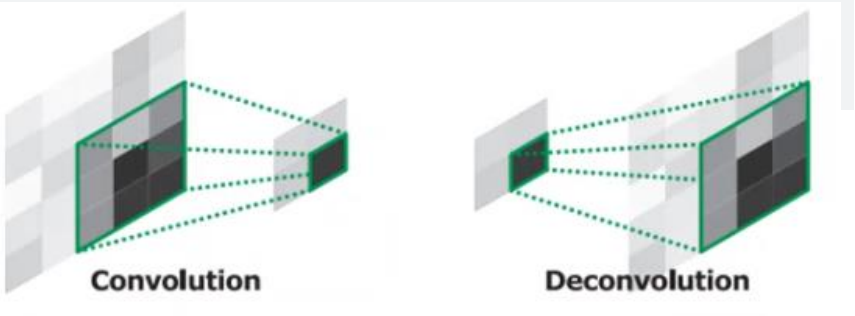
View source on GitHub

Transposed convolution layer (sometimes called Deconvolution).

Inherits From: `Conv2D`, `Layer`, `Module`

View aliases

```
tf.keras.layers.Conv2DTranspose(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    output_padding=None,
    data_format=None,
    dilation_rate=(1, 1),
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
```



https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose

tf.keras.layers.UpSampling2D

View source on GitHub

Upsampling layer for 2D inputs.

Inherits From: `Layer`, `Module`

View aliases

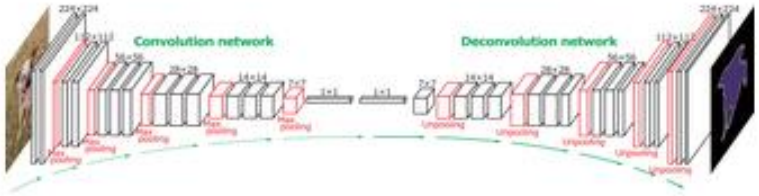
```
tf.keras.layers.UpSampling2D(
    size=(2, 2), data_format=None, interpolation='nearest', **kwargs
)
```

Used in the notebooks

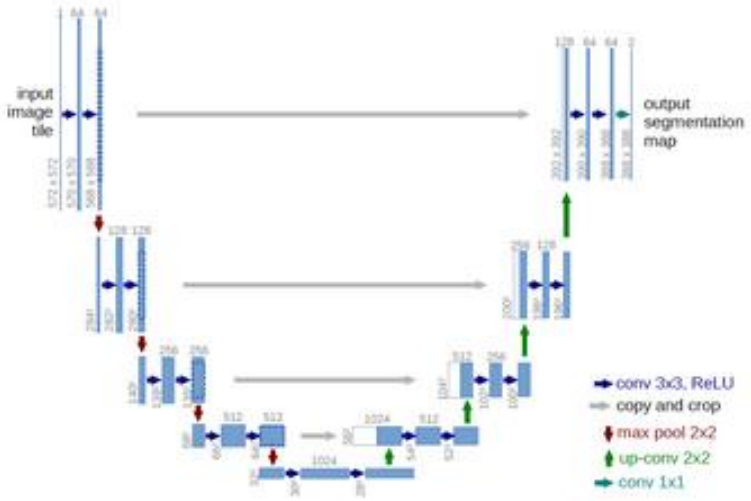
Args	
`size`	Int, or tuple of 2 integers. The upsampling factors for rows and columns.
`data_format`	A string, one of `channels_last` (default) or `channels_first`. The ordering of the dimensions in the inputs. `channels_last` corresponds to inputs with shape `(batch_size, height, width, channels)` while `channels_first` corresponds to inputs with shape `(batch_size, channels, height, width)`. It defaults to the `image_data_format` value found in your Keras config file at `~/.keras/keras.json`. If you never set it, then it will be "channels_last".
`interpolation`	A string, one of `"area"`, `"bicubic"`, `"bilinear"`, `"gaussian"`, `"lanczos3"`, `"lanczos5"`, `"mitchellcubic"`, `"nearest"`.

https://www.tensorflow.org/api_docs/python/tf/keras/layers/UpSampling2D

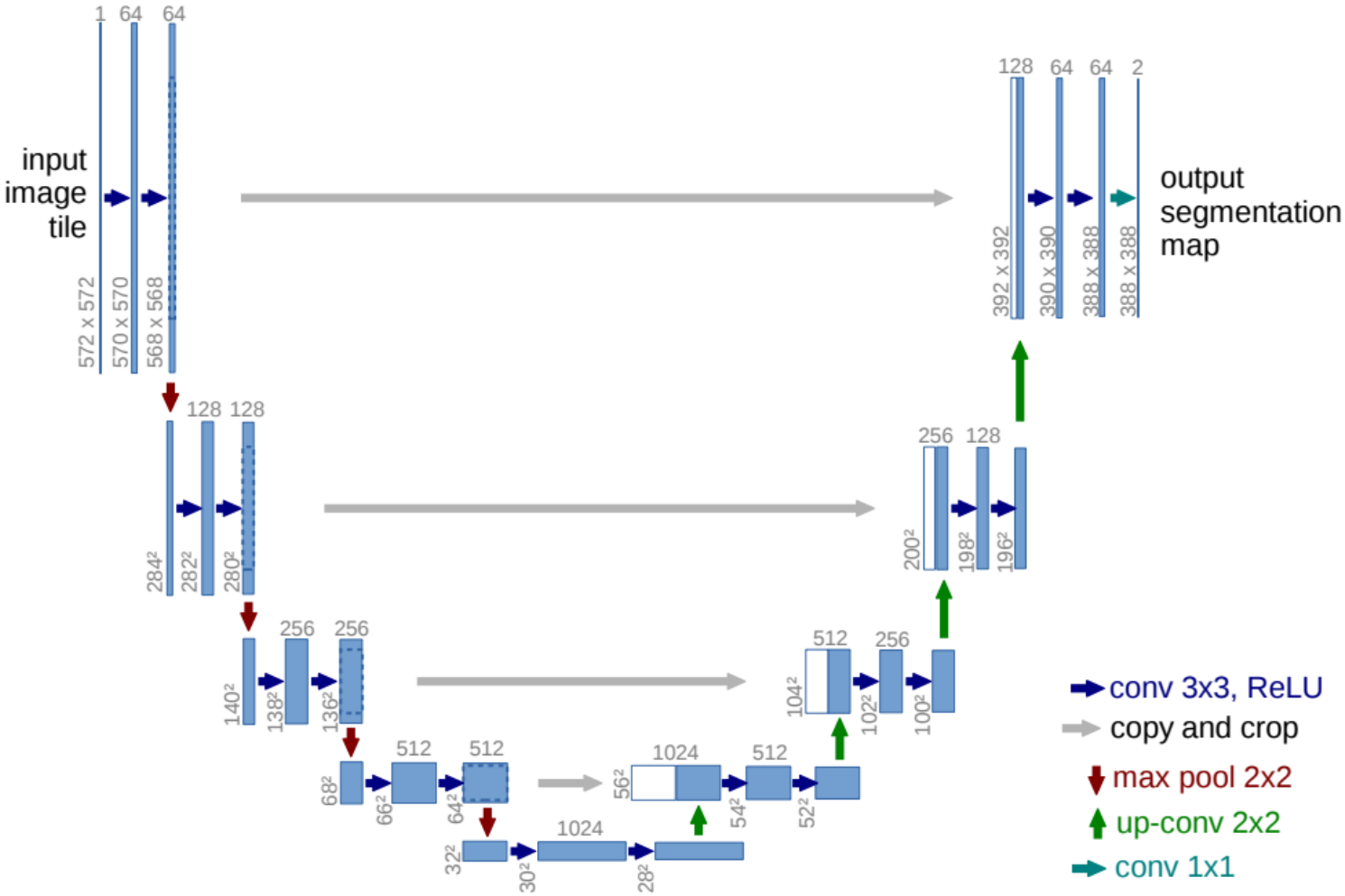
SegNet


Convolution network
Deconvolution network

UNet


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

U-net



https://arxiv.org/pdf/1505.04597.pdf

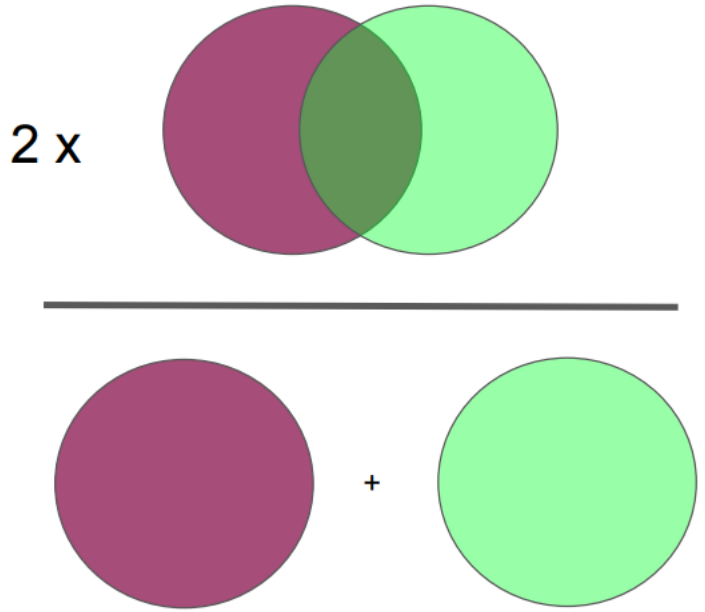
Metrics

Table 1. The three similarity coefficients

Similarity Coefficient (X,Y)	Actual Formula
Dice Coefficient	$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$
Cosine Coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} . \lvert Y \rvert^{1/2}}$
Jaccard Coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$

Dice Score

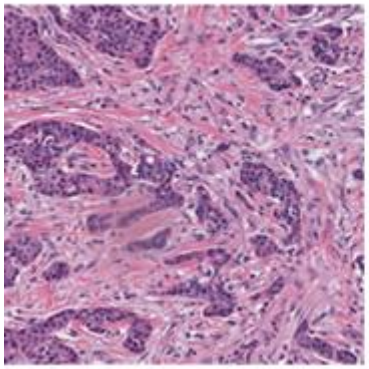
$$\text{Dice Score} = 2 \times \frac{\text{Area of Overlap}}{\text{Combined Area}}$$

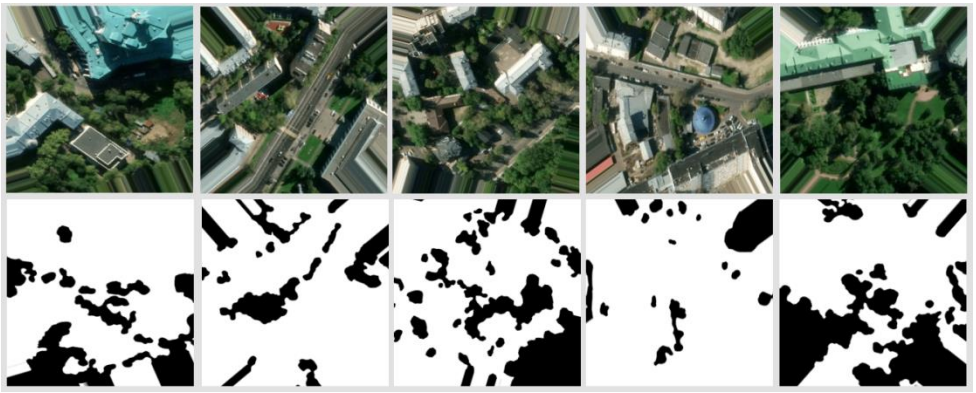


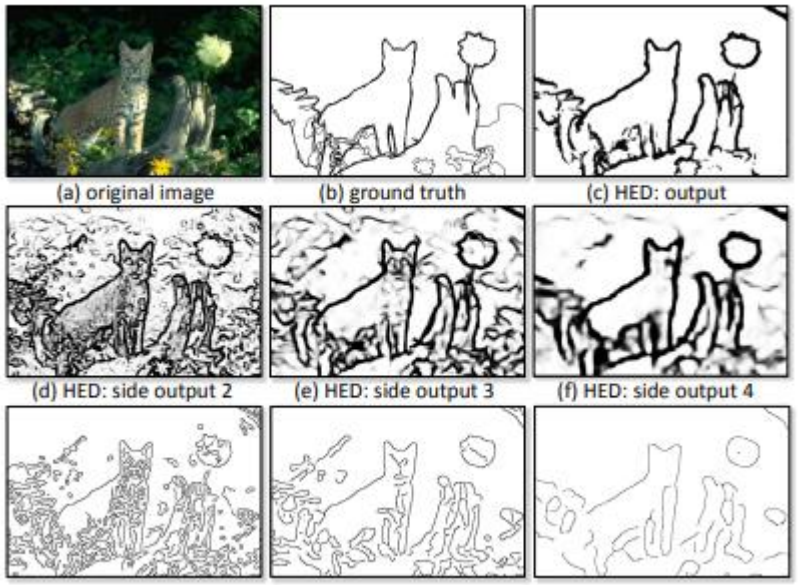
https://www.coursera.org/learn/advanced-computer-vision-with-tensorflow/home/info

https://www.kaggle.com/code/yerramvarun/understanding-dice-coefficient

Practical applications



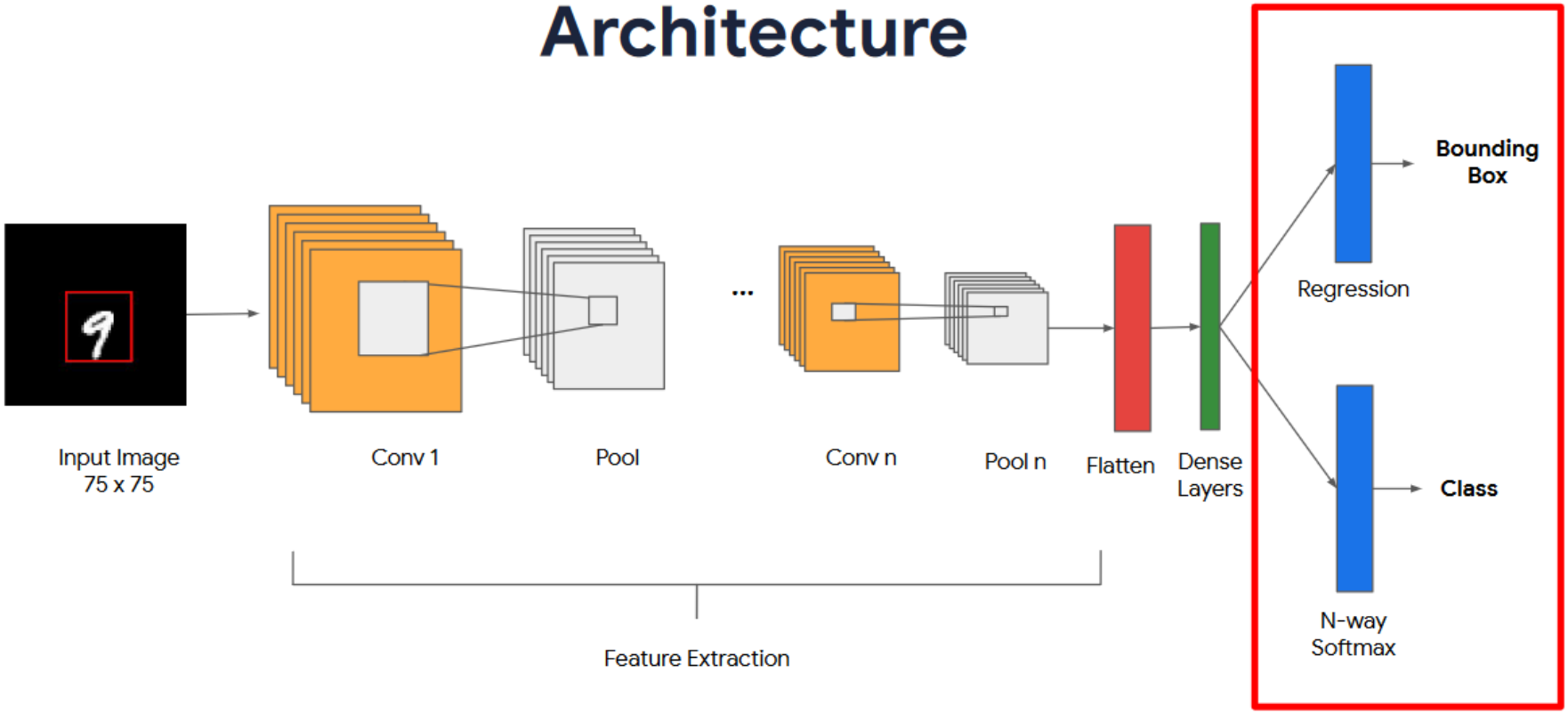




Classification + Localization

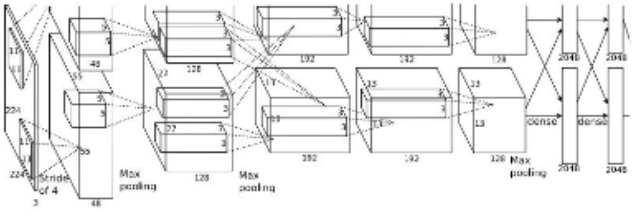
CAT

Convolutional Neural Networks Architecture



Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Multitask Loss

+

Loss

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

Treat localization as a regression problem!

https://www.youtube.com/watch?v=[illegible]WoltGTWV54&index=14&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&t=0s

Intersection Over Union

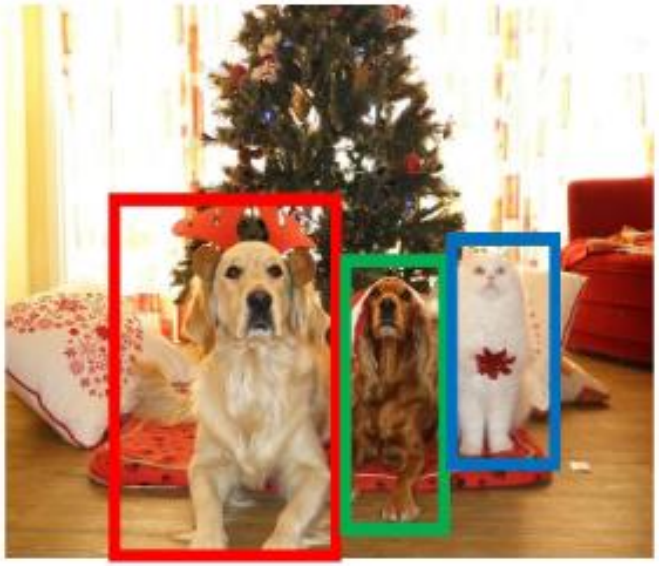
$$IOU = \frac{\text{Intersection}}{\text{Union}}$$

https://www.coursera.org/learn/advanced-computer-vision-with-tensorflow/home/info

Intersection Over Union

$$IOU = \frac{\text{Intersection}}{\text{Union}}$$

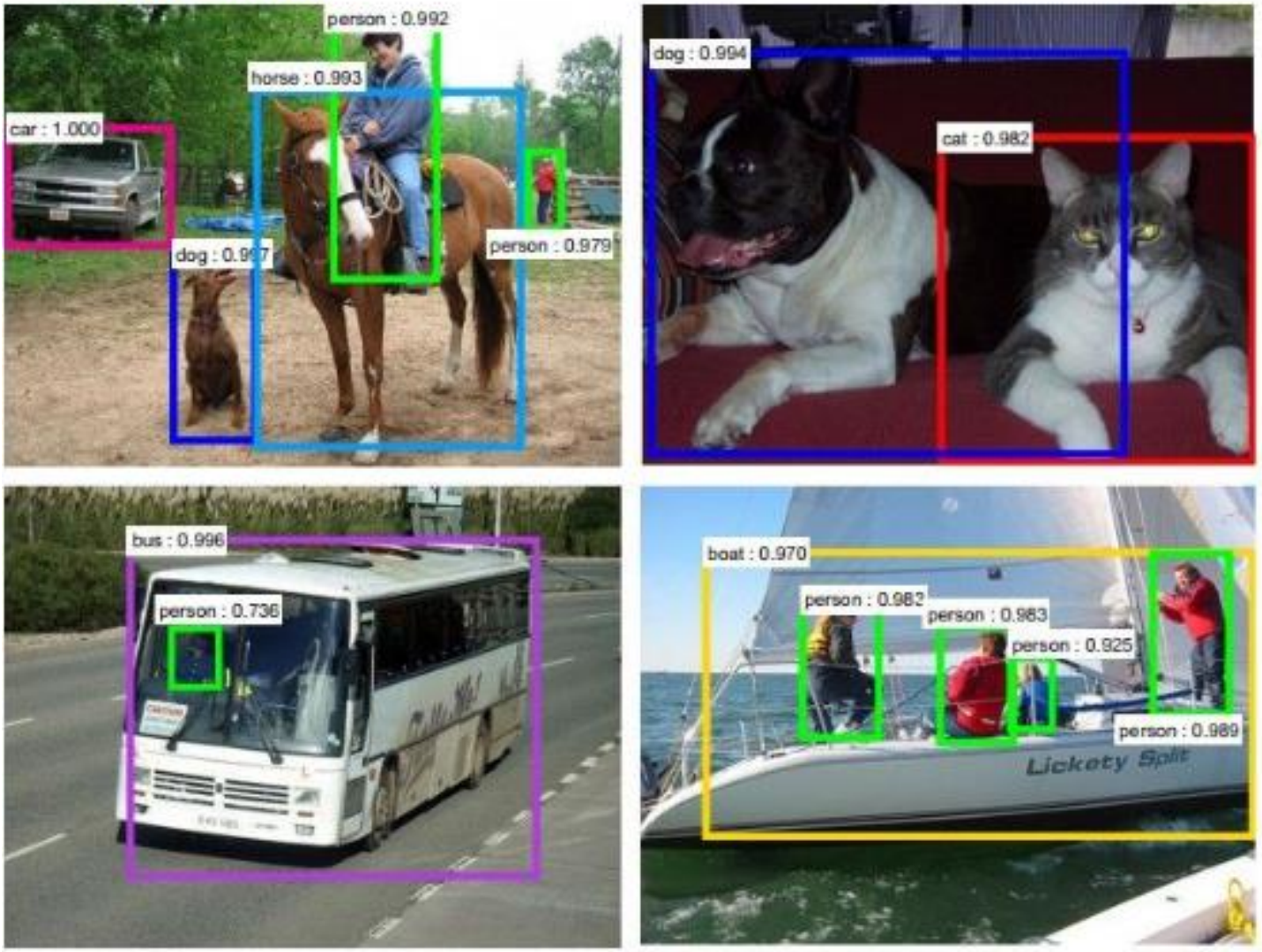
https://www.coursera.org/learn/convolutional-neural-networks/home/week/3



DOG, DOG, CAT

Object Detection

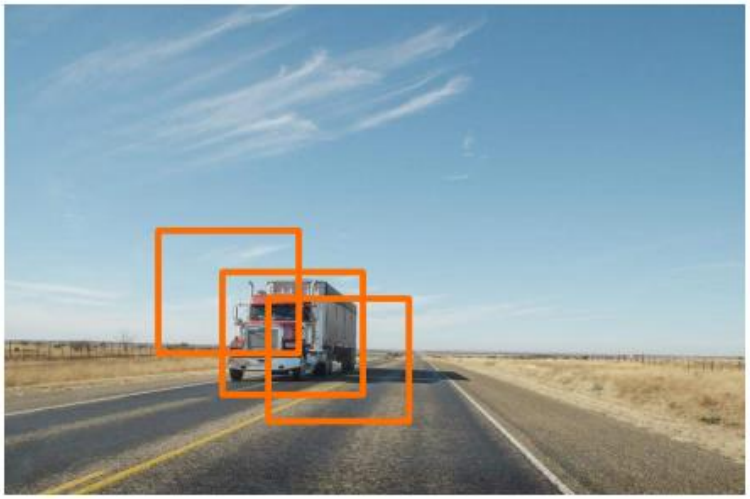
Detection


person : 0.992
horse : 0.993
car : 1.000
dog : 0.9
person : 0.979
dog : 0.994
cat : 0.982
bus : 0.996
person : 0.736
boat : 0.970
person : 0.982
person : 0.983
person : 0.925
person : 0.989
Lickety Split

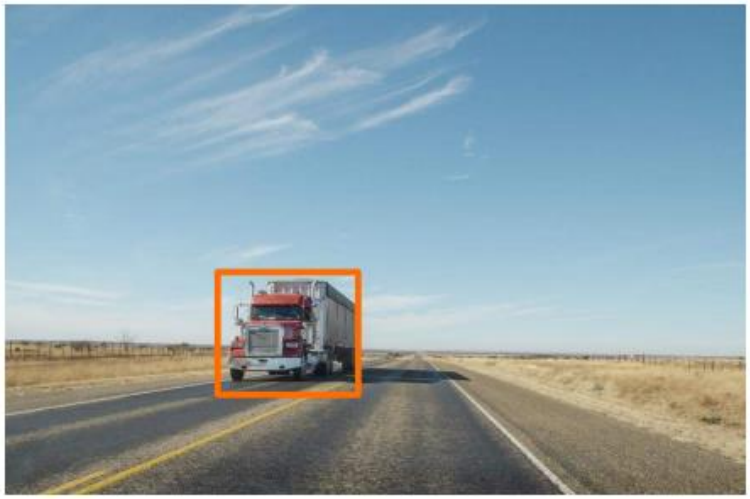
Two stages to object detection

- Region proposal
- Object detection and classification

Non-maximum suppression (NMS)

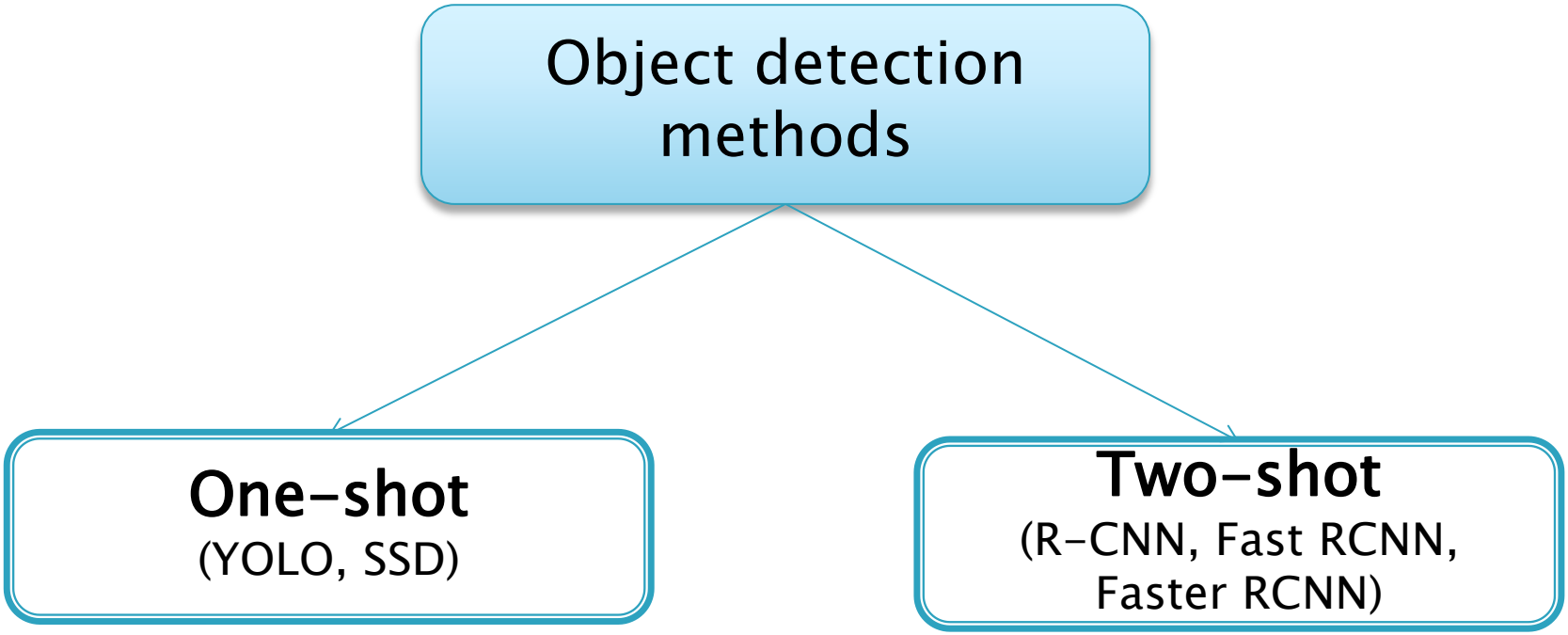


NMS

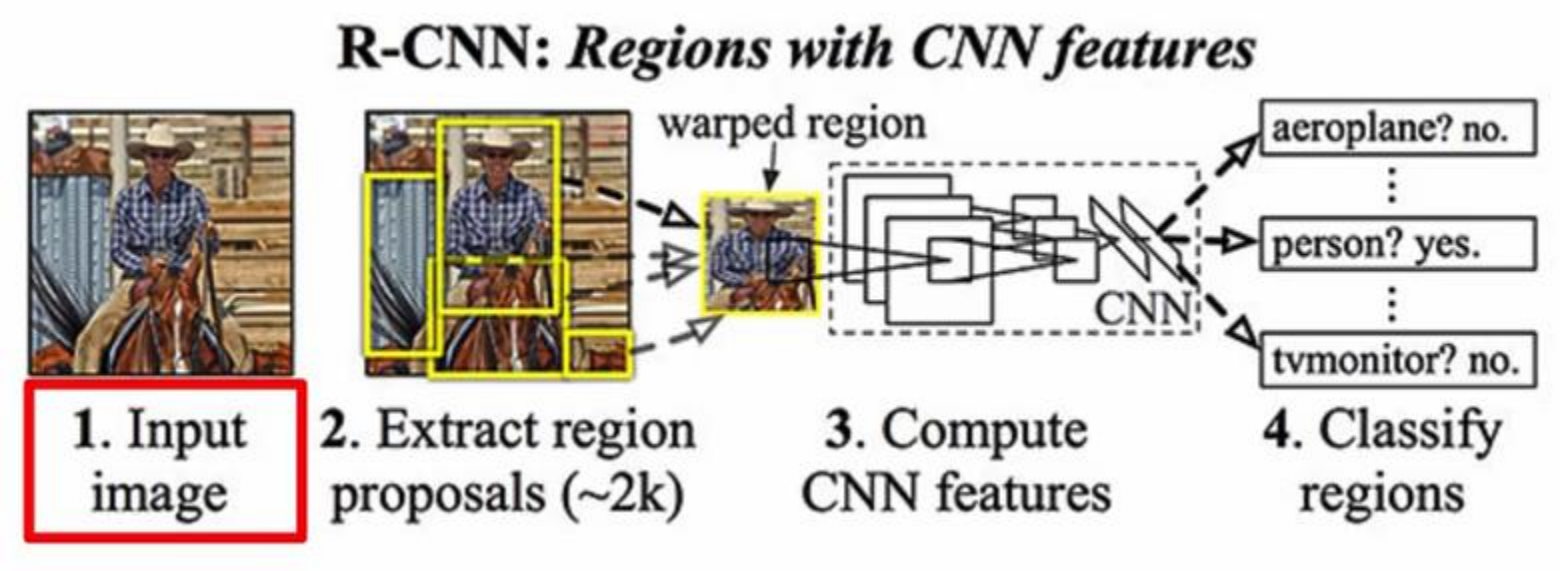


Before NMS

After NMS


Object detection methods
One-shot
(YOLO, SSD)
Two-shot
(R-CNN, Fast RCNN, Faster RCNN)

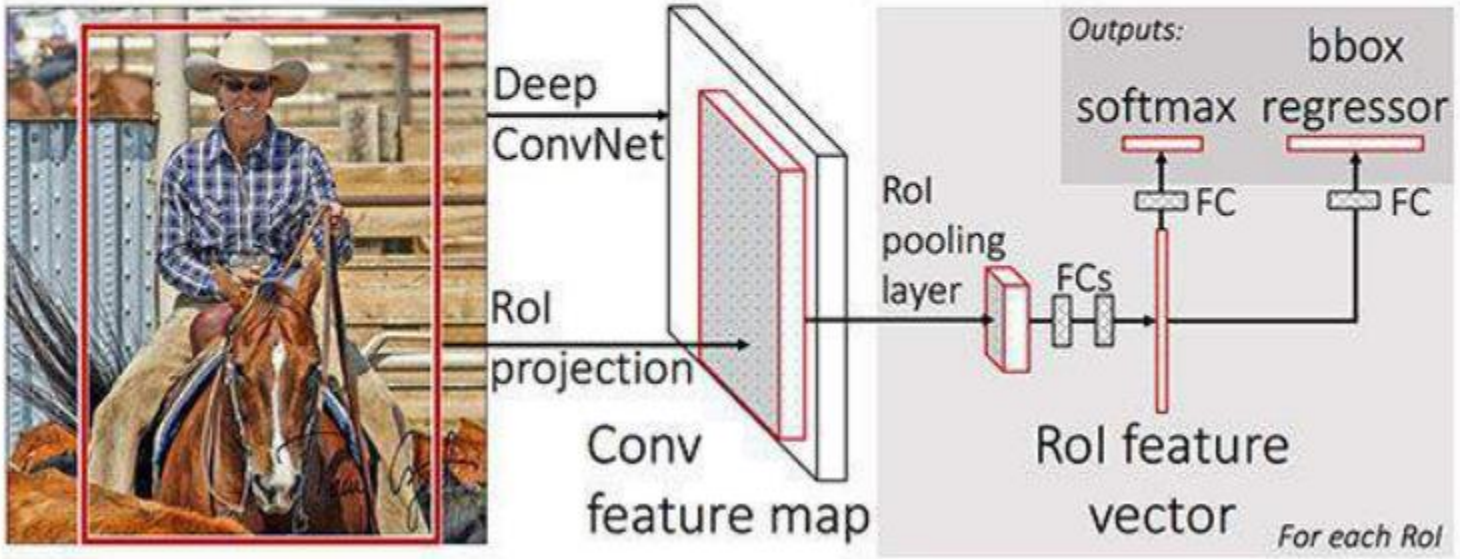
R-CNN



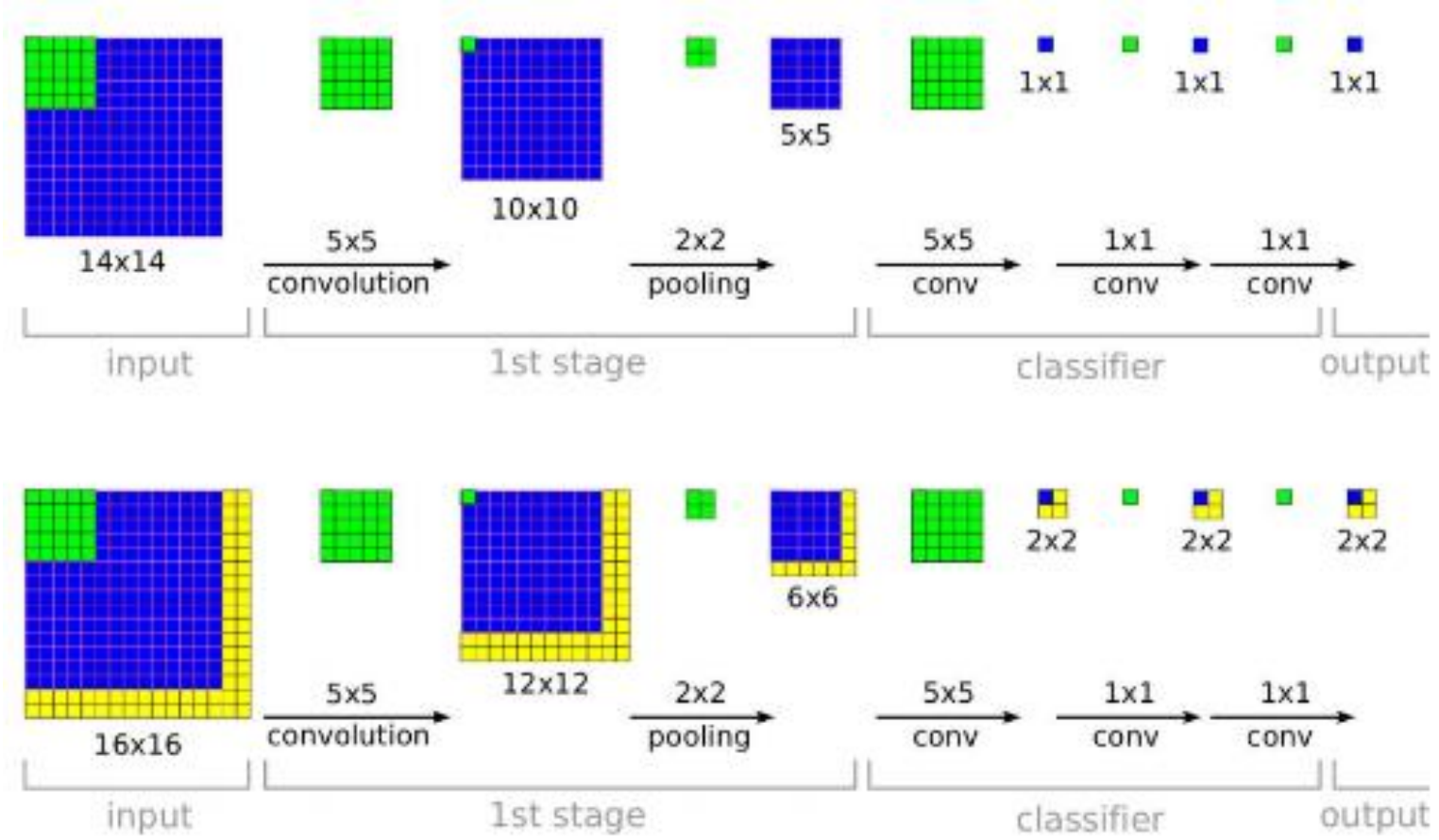
https://arxiv.org/abs/1311.2524

https://www.koen.me/research/pub/uijlings-ijcv2013-draft.pdf

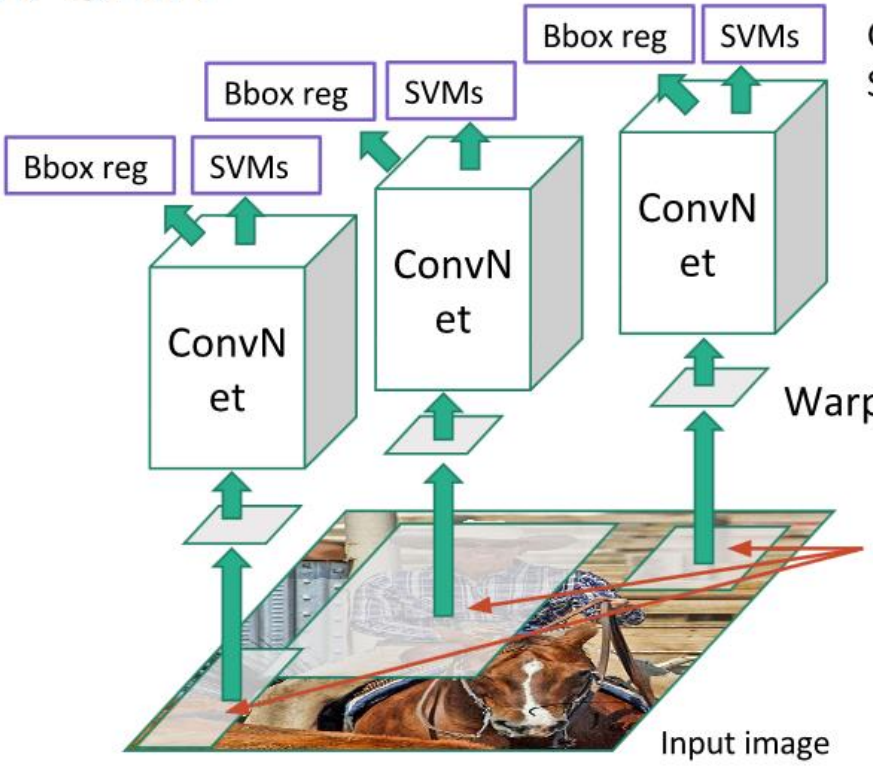
Fast R-CNN



https://arxiv.org/abs/1504.08083


14x14
5x5
convolution
10x10
2x2
pooling
5x5
1x1
1x1
5x5
conv
1x1
conv
1x1
conv
input
1st stage
classifier
output
16x16
5x5
convolution
12x12
2x2
pooling
6x6
2x2
2x2
2x2
5x5
conv
1x1
conv
1x1
conv
input
1st stage
classifier
output

R-CNN



Linear Regression for bounding box offsets

Classify regions with SVMs

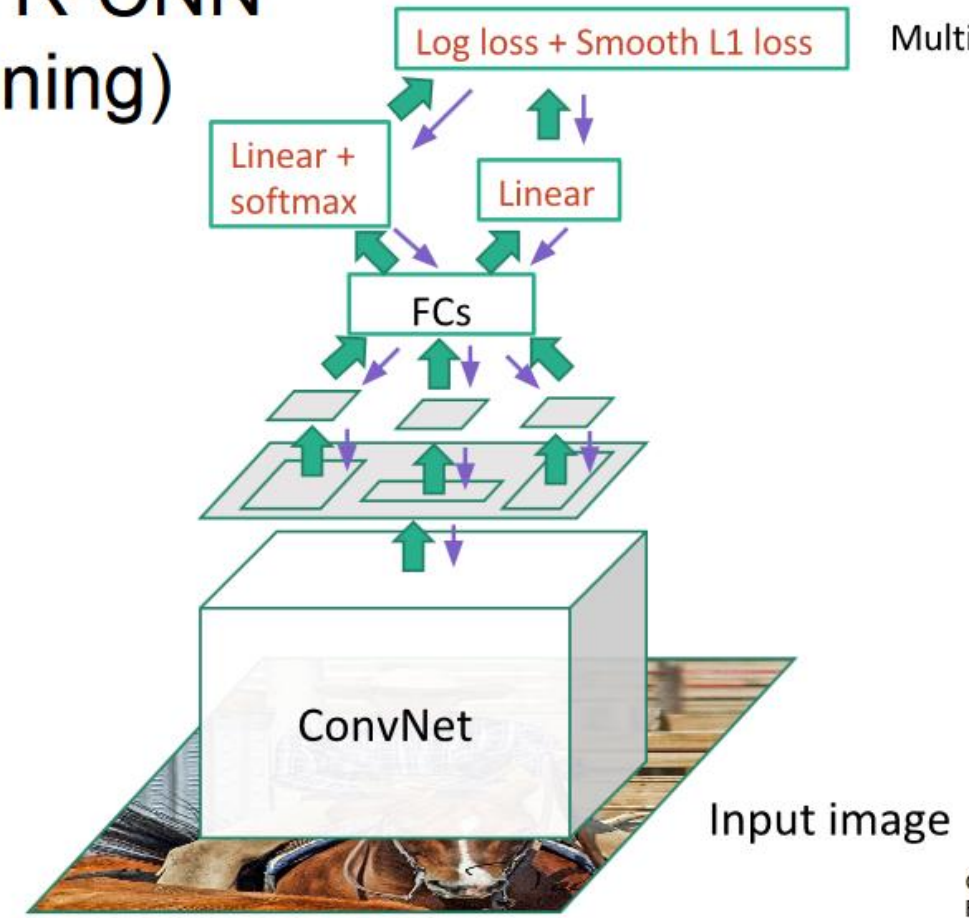
Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

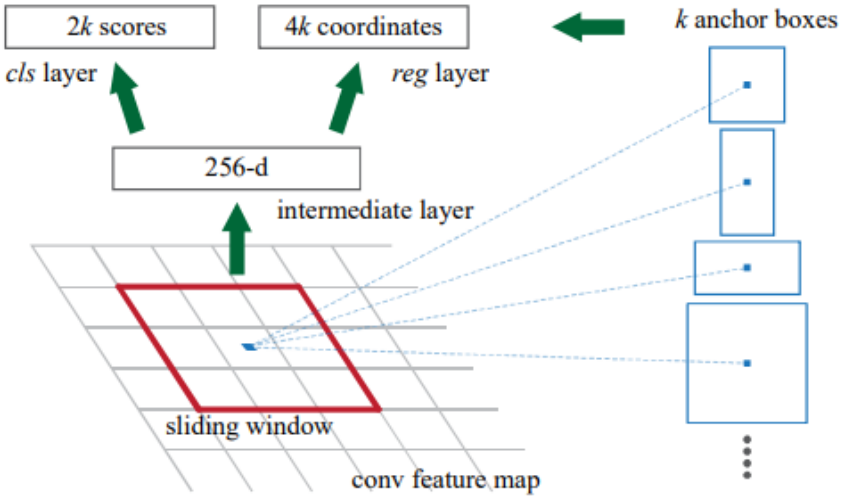
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

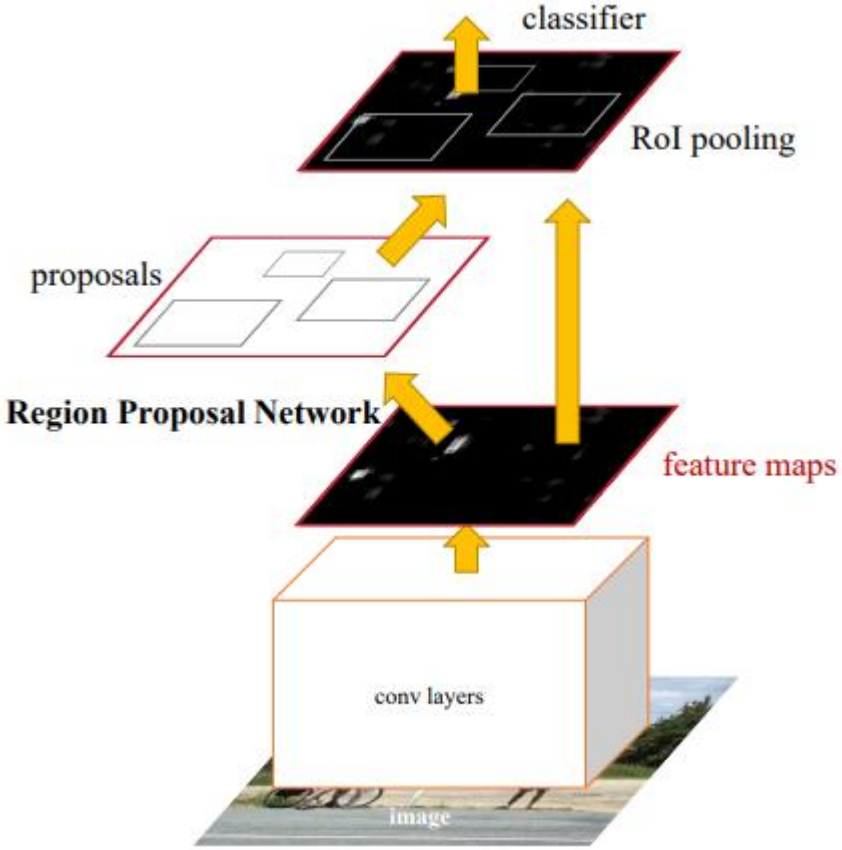
Fast R-CNN (Training)



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

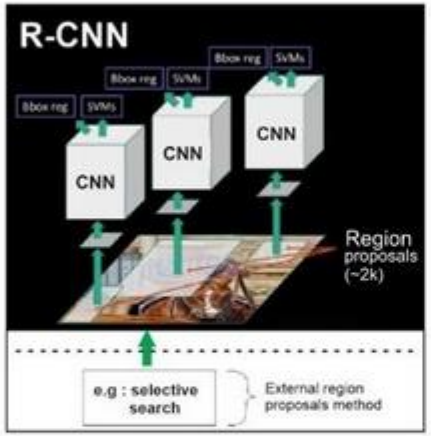
Faster R-CNN

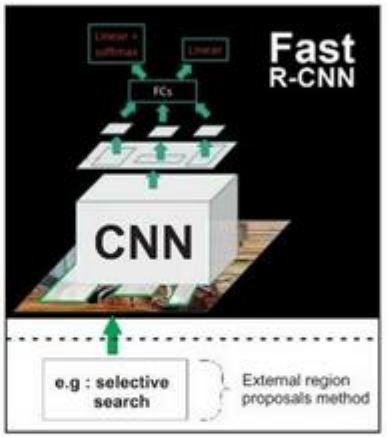


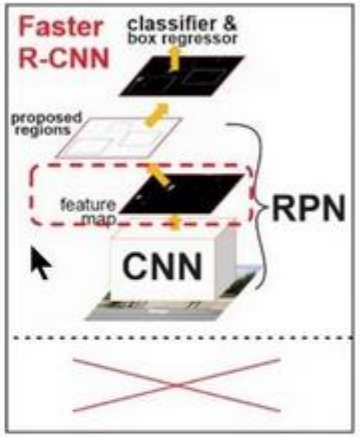


https://arxiv.org/abs/1506.01497

Two shot: performance





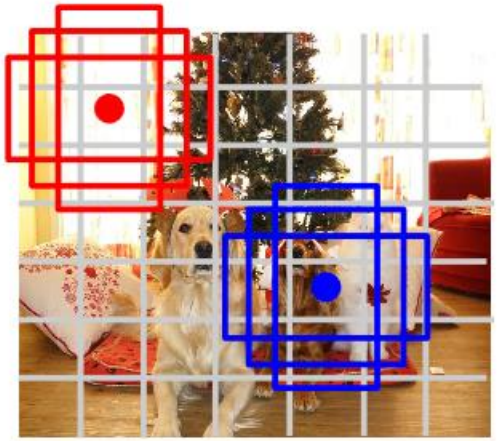


	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

https://slides.com/vladimiriglovikov/title-texttitle-text-17#/0/24

Detection without Proposals: YOLO / SSD

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

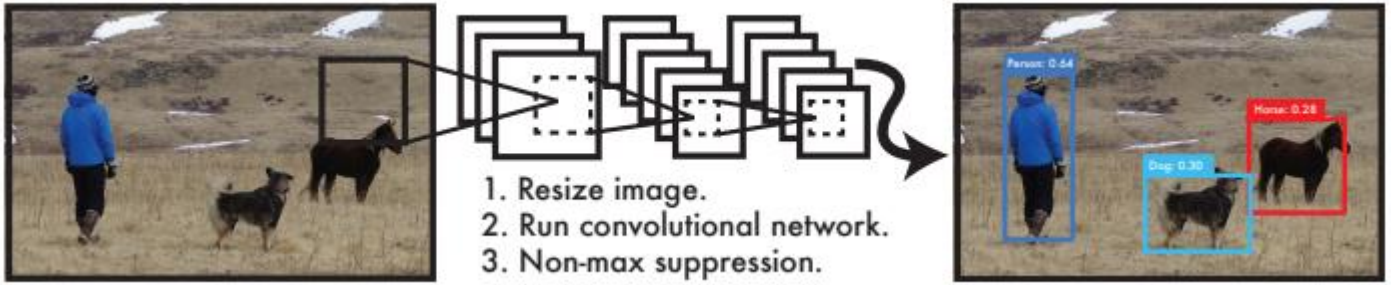
Within each grid cell:

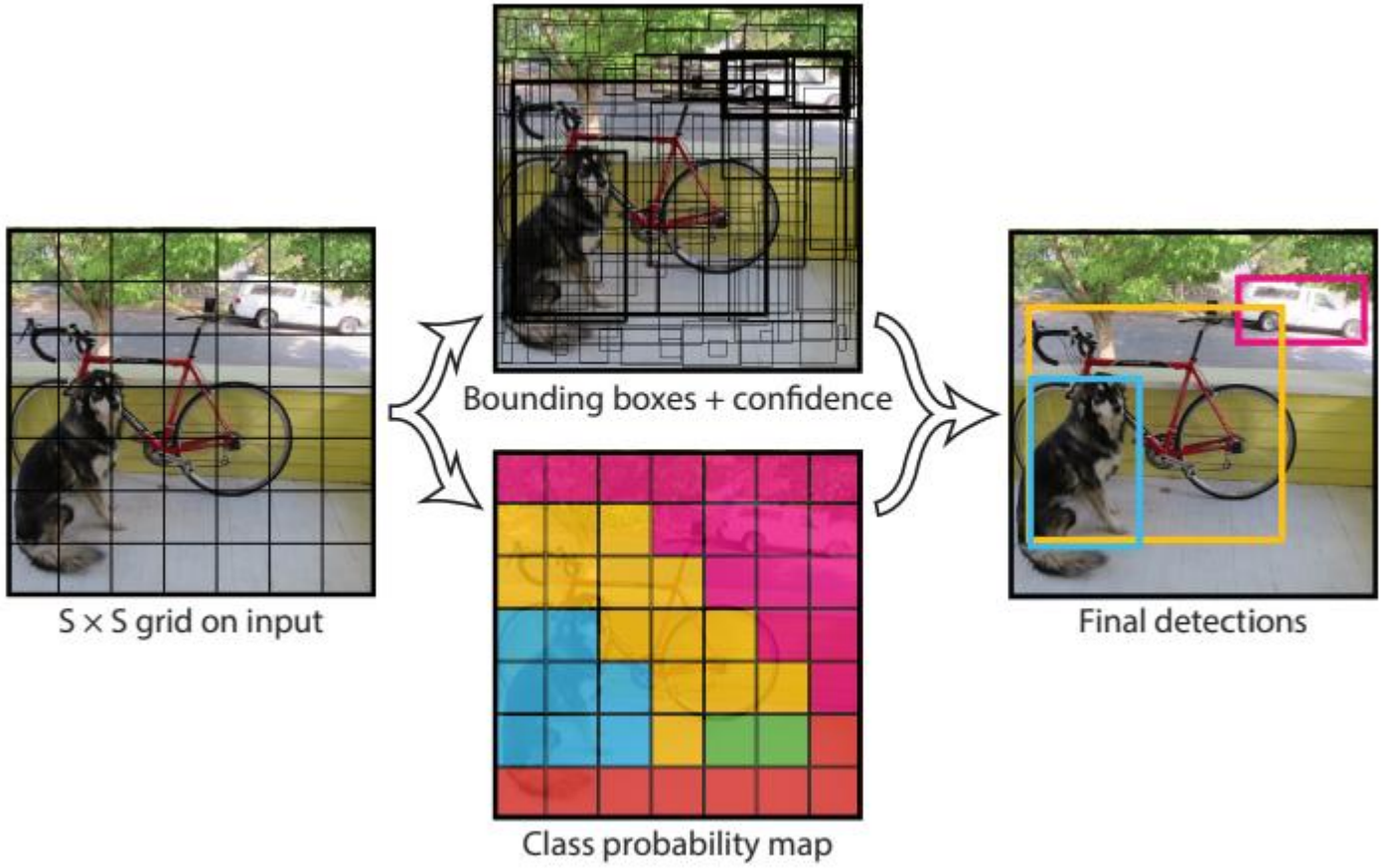
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

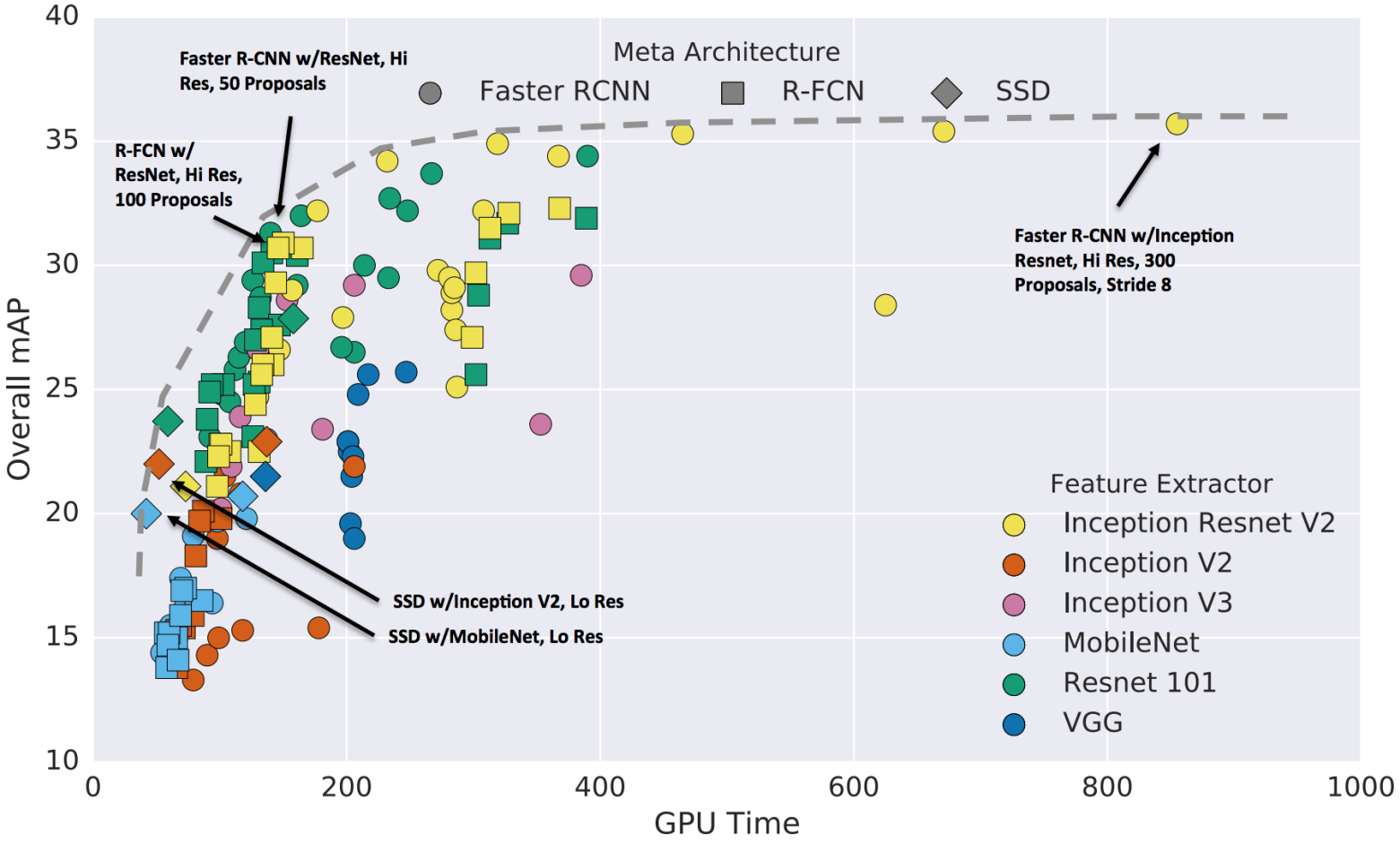
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

https://www.youtube.com/watch?v=[illegible]WoItGTWV54&index=14&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&t=0s





https://arxiv.org/abs/1506.02640



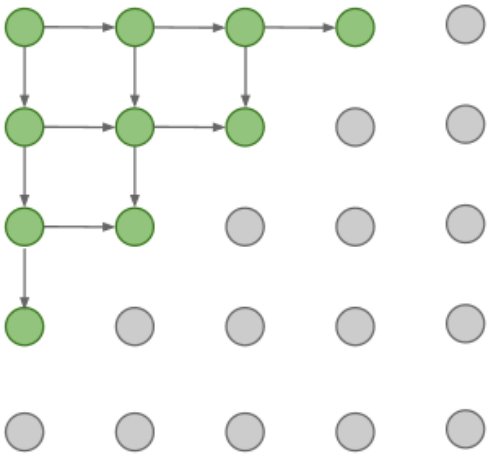
https://arxiv.org/abs/1611.10012

Generative models

PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)



https://www.youtube.com/wat[illegible]WoltGTWV54&index=14&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&t=0s

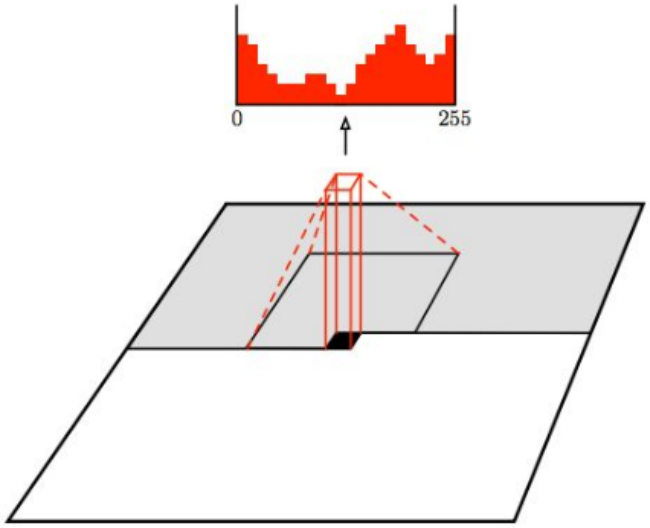
PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training is faster than PixelRNN
(can parallelize convolutions since context region values known from training images)

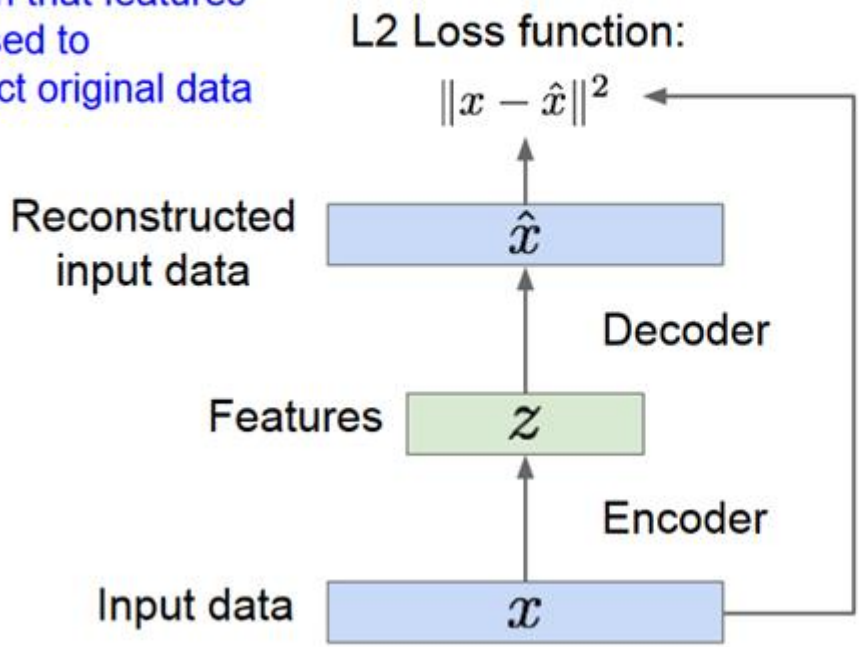
Generation must still proceed sequentially
=> still slow

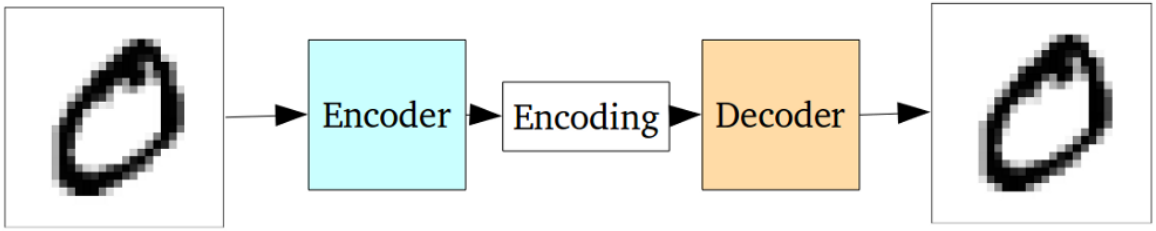


https://www.youtube.com/watch?v=[illegible]WoItGTWV54&index=14&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&t=0s

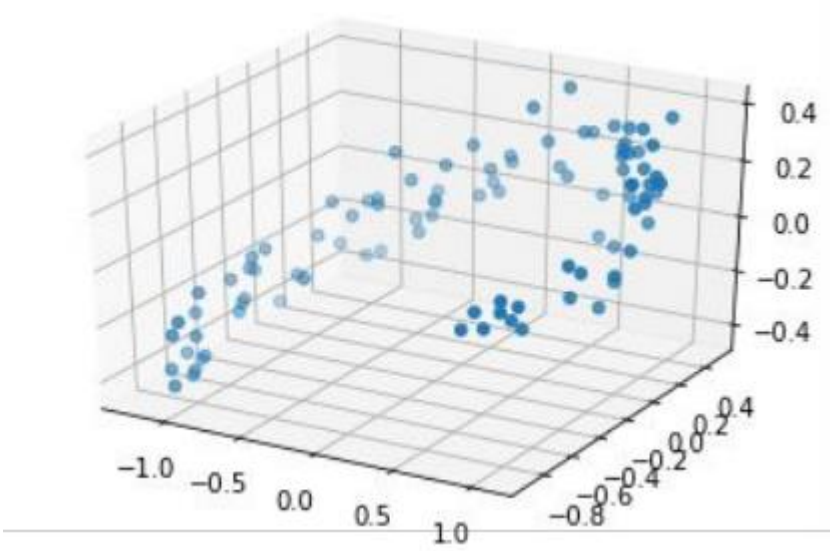
Autoencoders

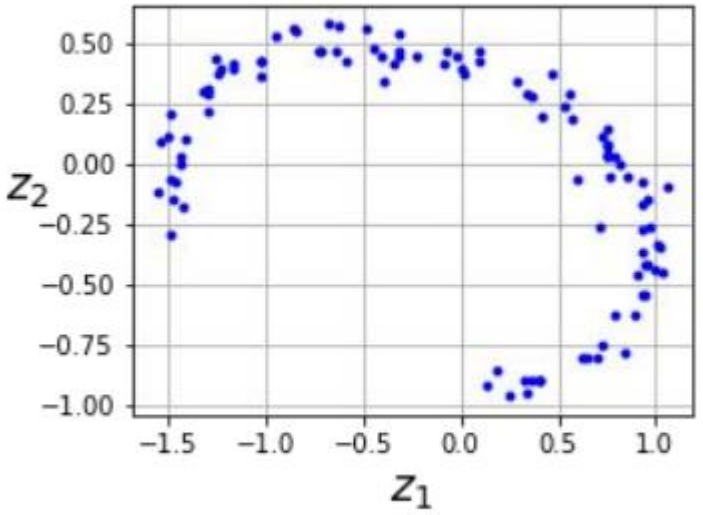
Train such that features can be used to reconstruct original data



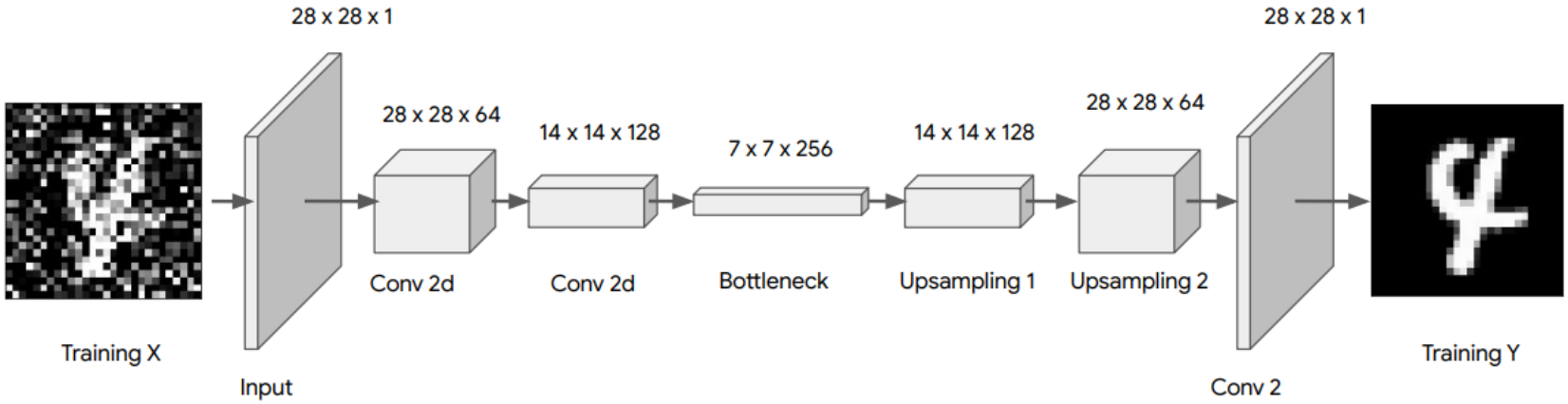


Dimensionality reduction



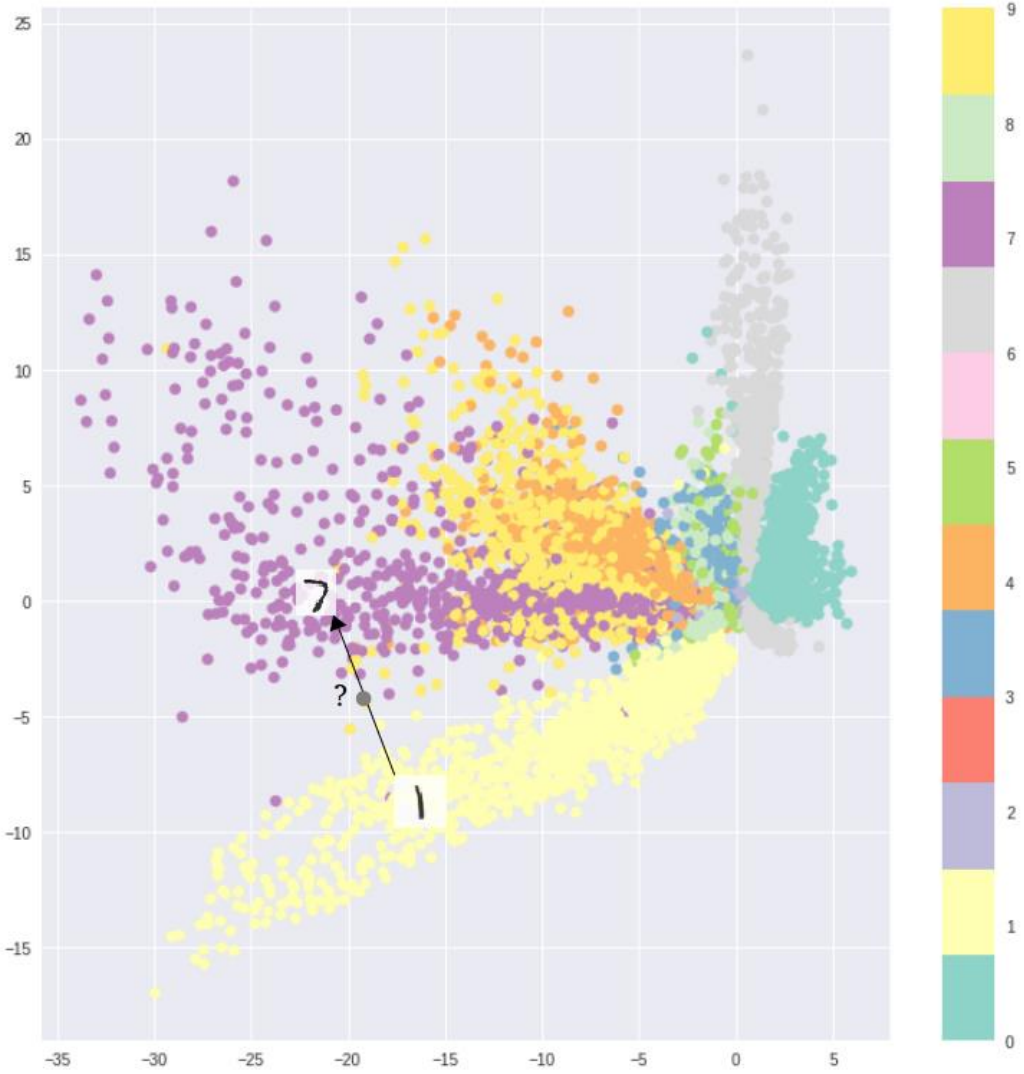


Noise reduction



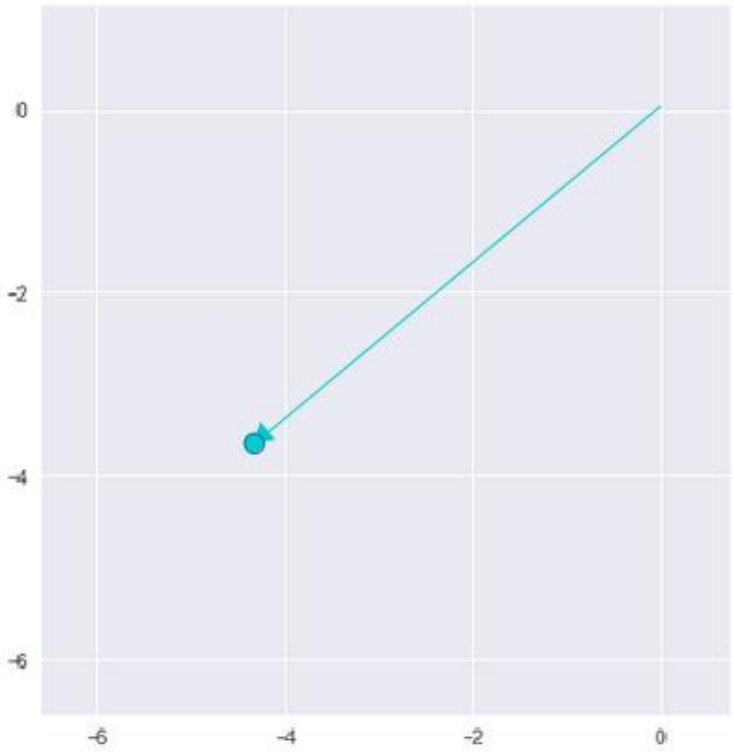
https://www.coursera.org/learn/generative-deep-learning-with-tensorflow/home/info

Autoencoder as generative model

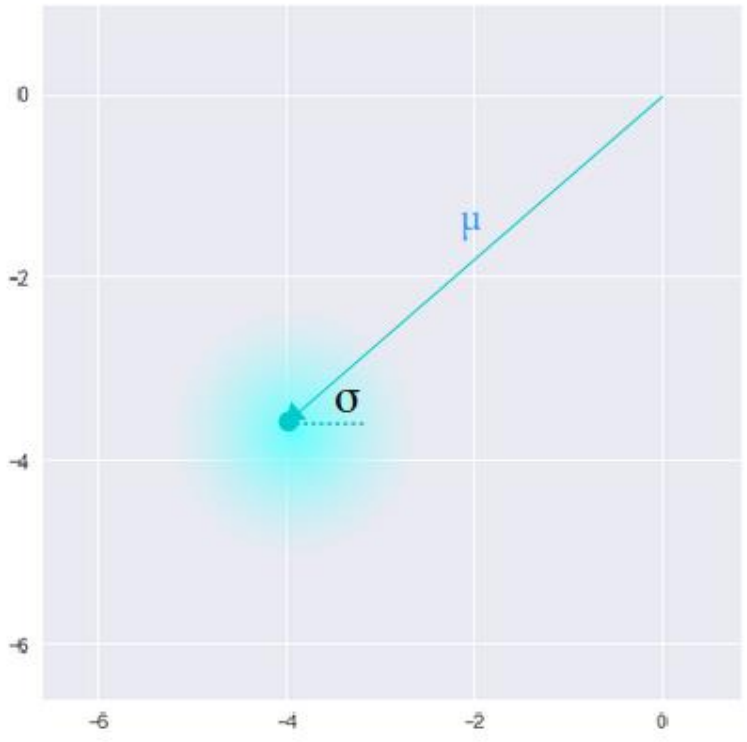


https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/

Variational Autoencoder (VAE)

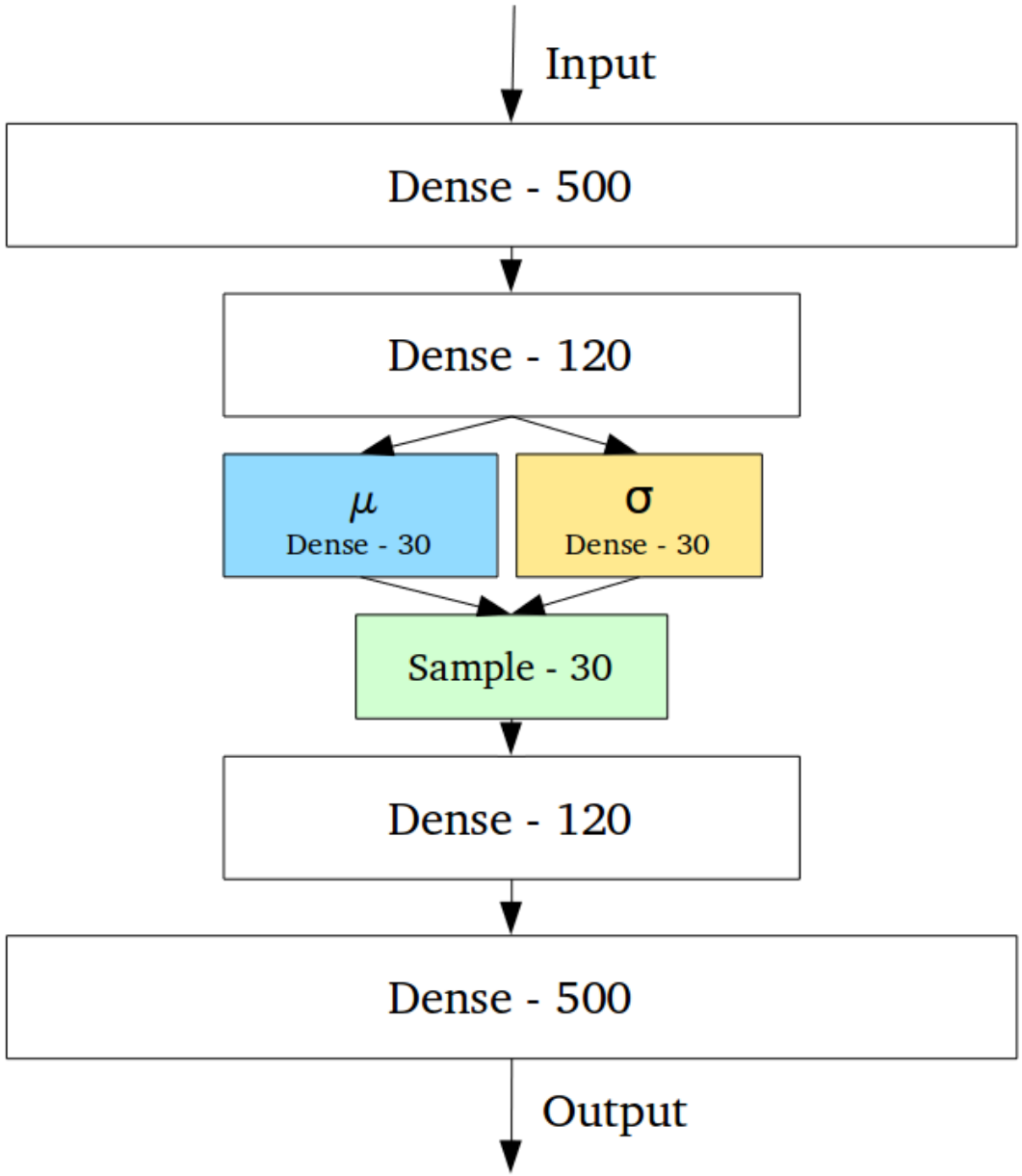


Standard Autoencoder
(direct encoding coordinates)

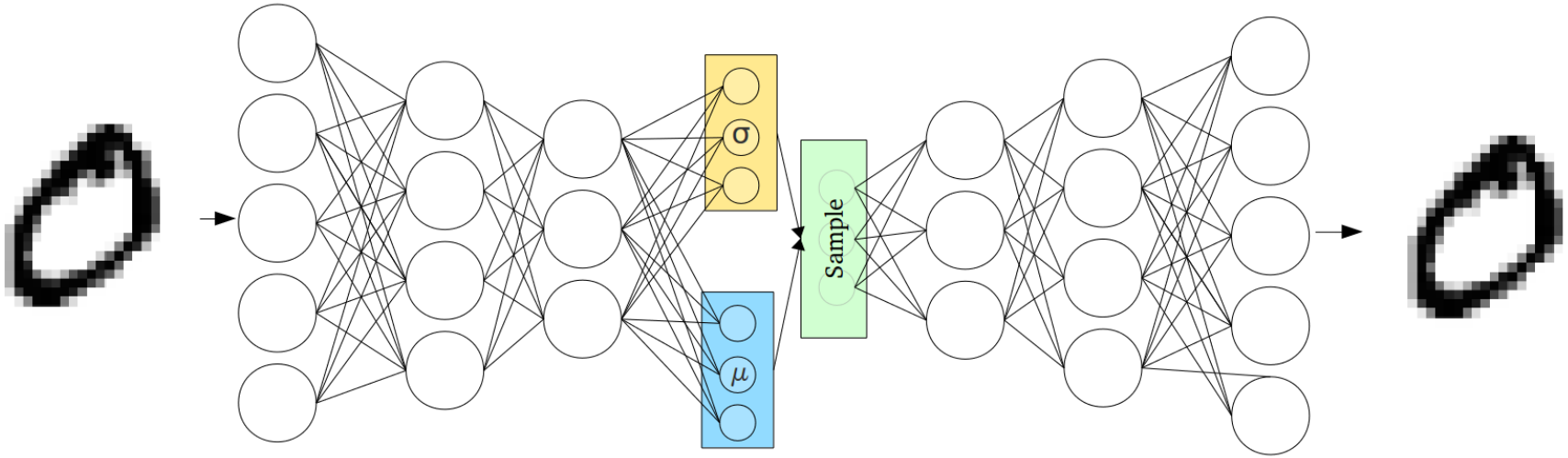


Variational Autoencoder
(μ and σ initialize a probability distribution)

https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/



https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/



Output
μ [0.1, 1.2, 0.2, 0.8,...]

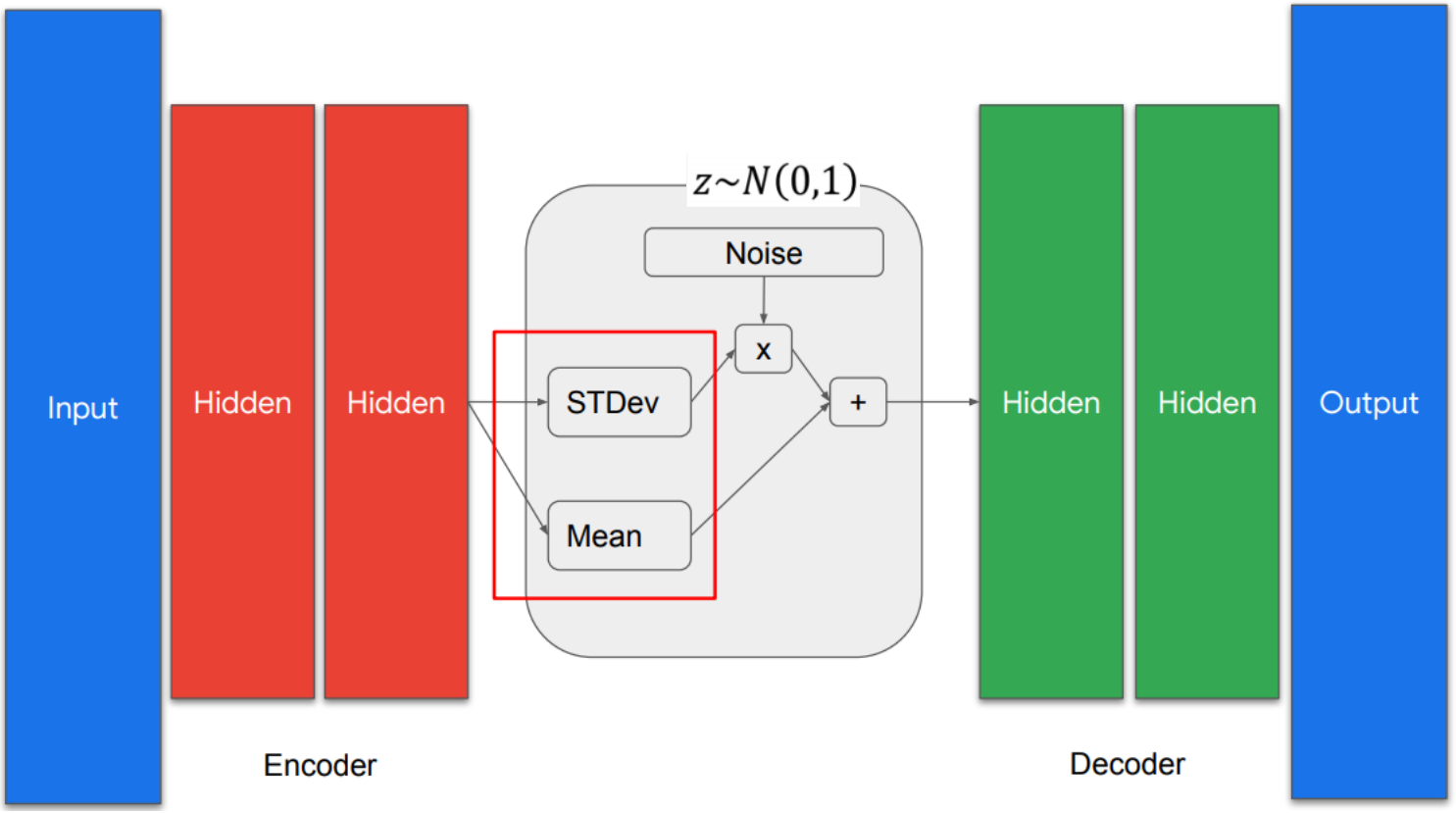
Output
$\boldsymbol{\sigma}$ [0.2, 0.5, 0.8, 1.3,...]

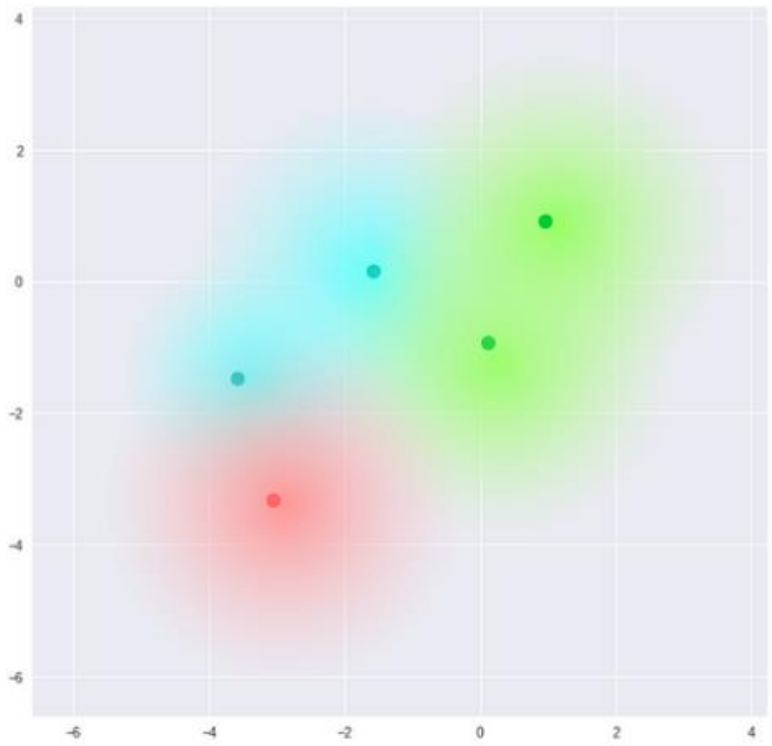
Intermediate
$\mathbf{X}$ $[X_1 \sim N(0.1, 0.2^2), X_2 \sim N(1.2, 0.5^2), X_3 \sim N(0.2, 0.8^2), X_4 \sim N(0.8, 1.3^2), \ldots]$

↓ sample

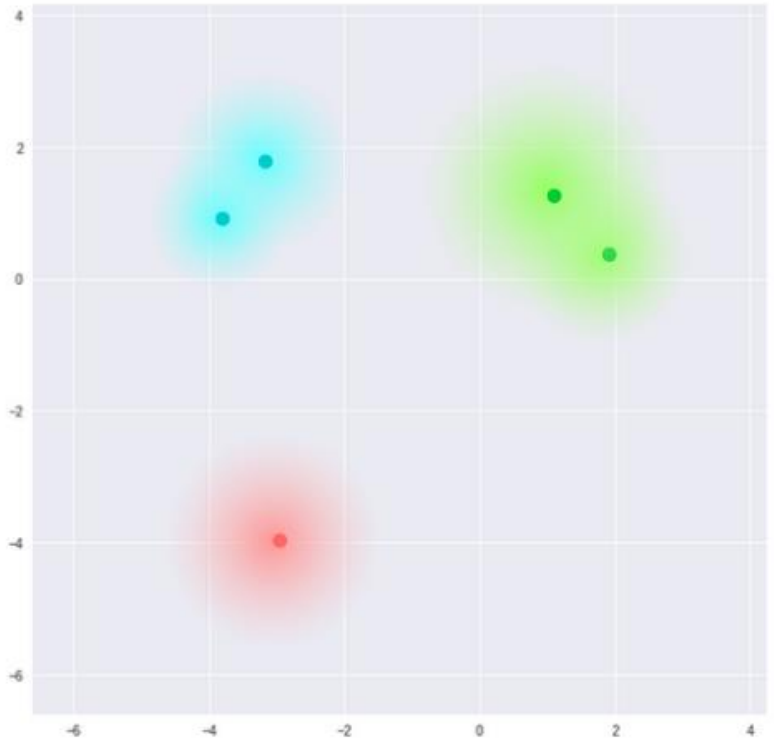
Sampled vector [0.28, 1.65, 0.92, 1.98,...]

https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/


Input
Hidden
Hidden
$z \sim N(0,1)$
Noise
x
STDev
+
Mean
Hidden
Hidden
Output
Encoder
Decoder



What we require

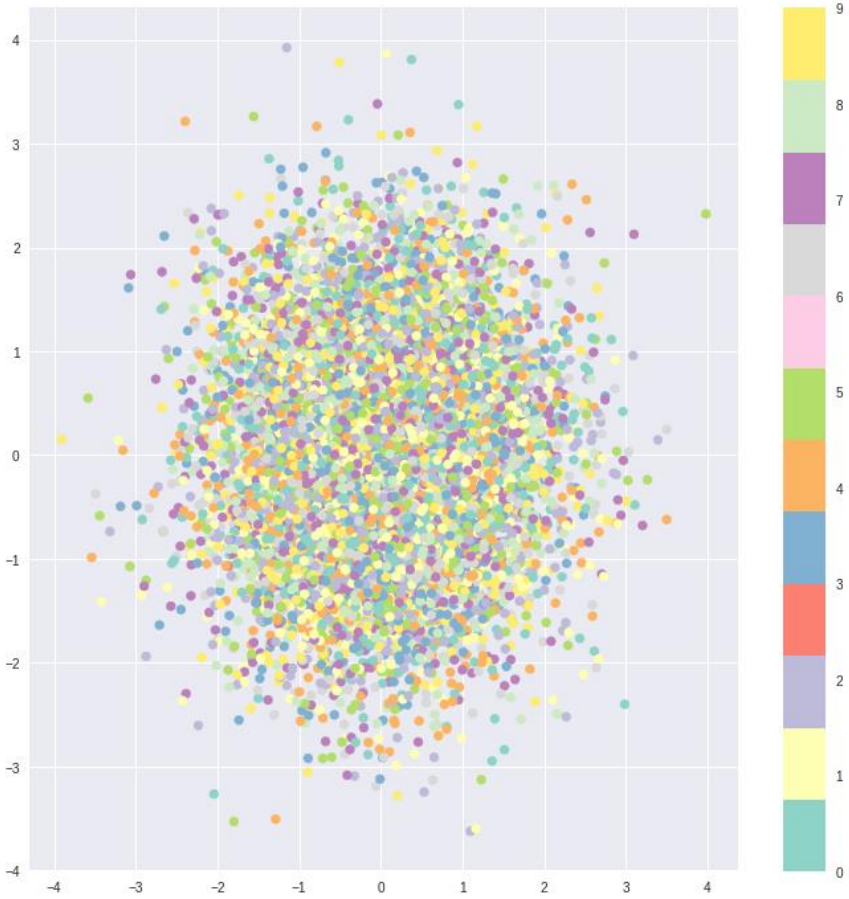


What we may inadvertently end up with

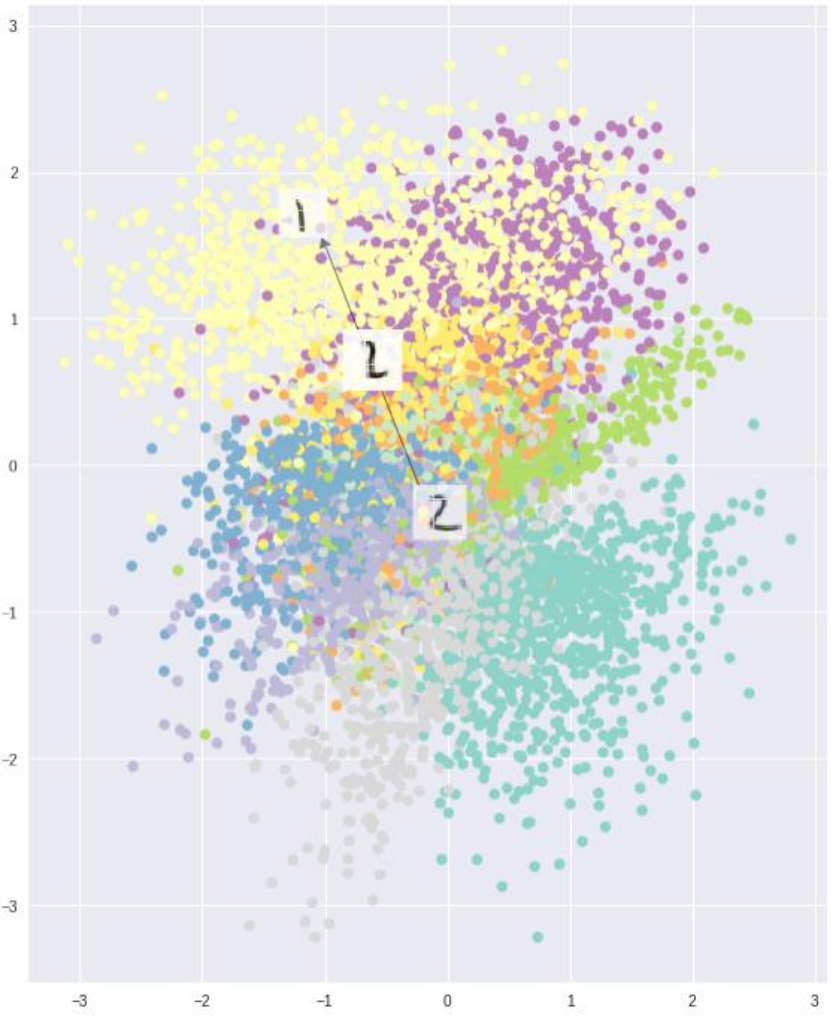
https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/

KL (Kullback-Leibler) divergence

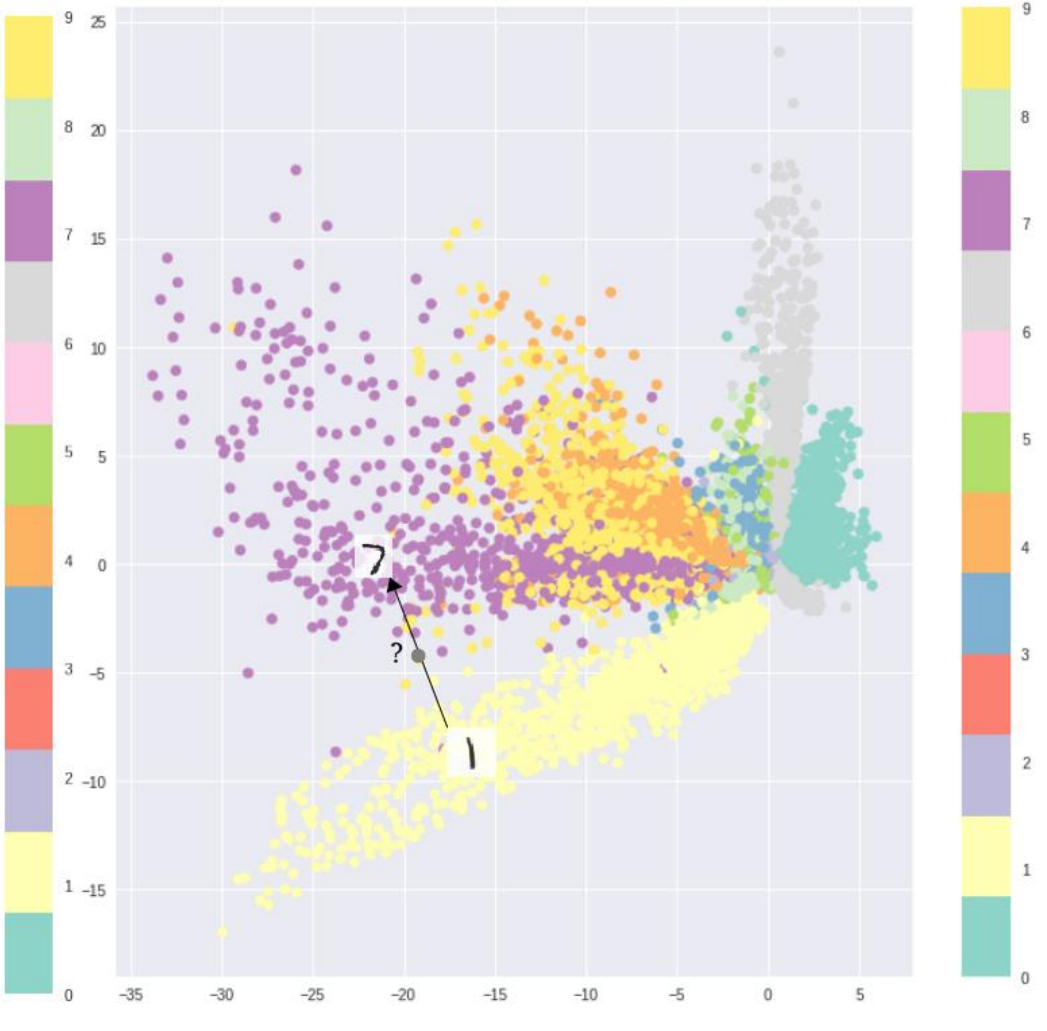
$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$



https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/



VAE



AE

https://neurohive.io/ru/osnovy-data-science/variacionnyj-avtojenkoder-vae/

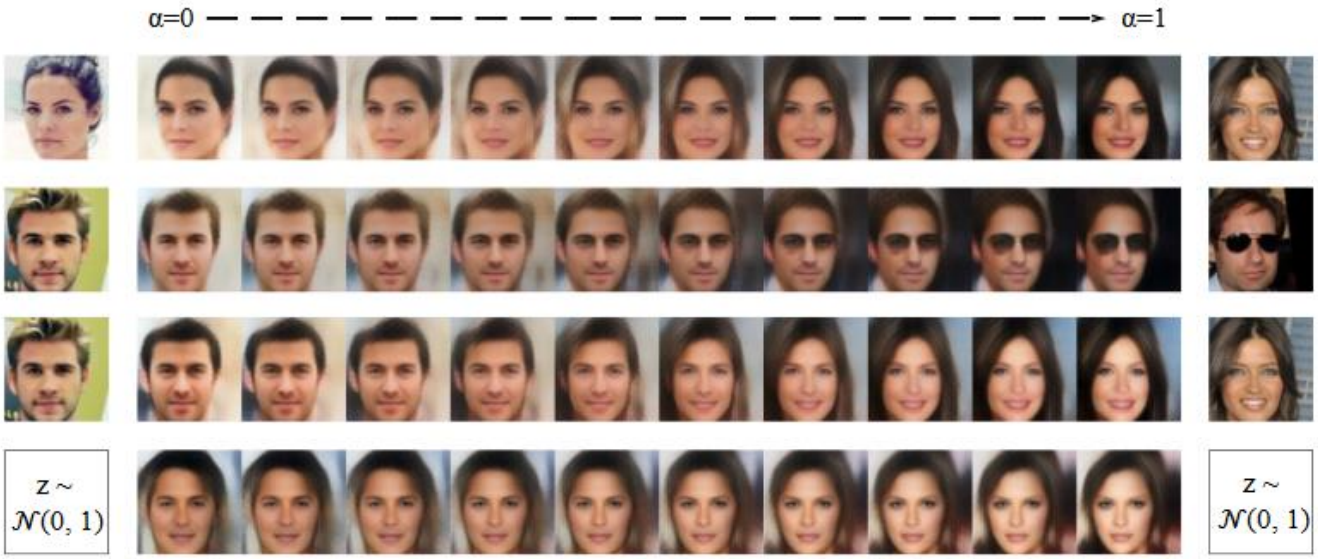
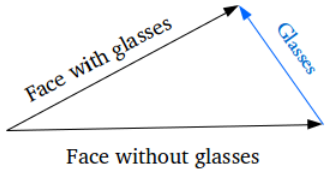


Figure 5. Linear interpolation for latent vector. Each row is the interpolation from left latent vector z_{left} to right latent vector z_{right}. e.g. $(1-\alpha)z_{left} + \alpha z_{right}$. The first row is the transition from a non-smiling woman to a smiling woman, the second row is the transition from a man without eyeglass to a man with eyeglass, the third row is the transition from a man to a woman, and the last row is the transition between two fake faces decoded from $z \sim \mathcal{N}(0, 1)$.

https://arxiv.org/pdf/1610.00291.pdf



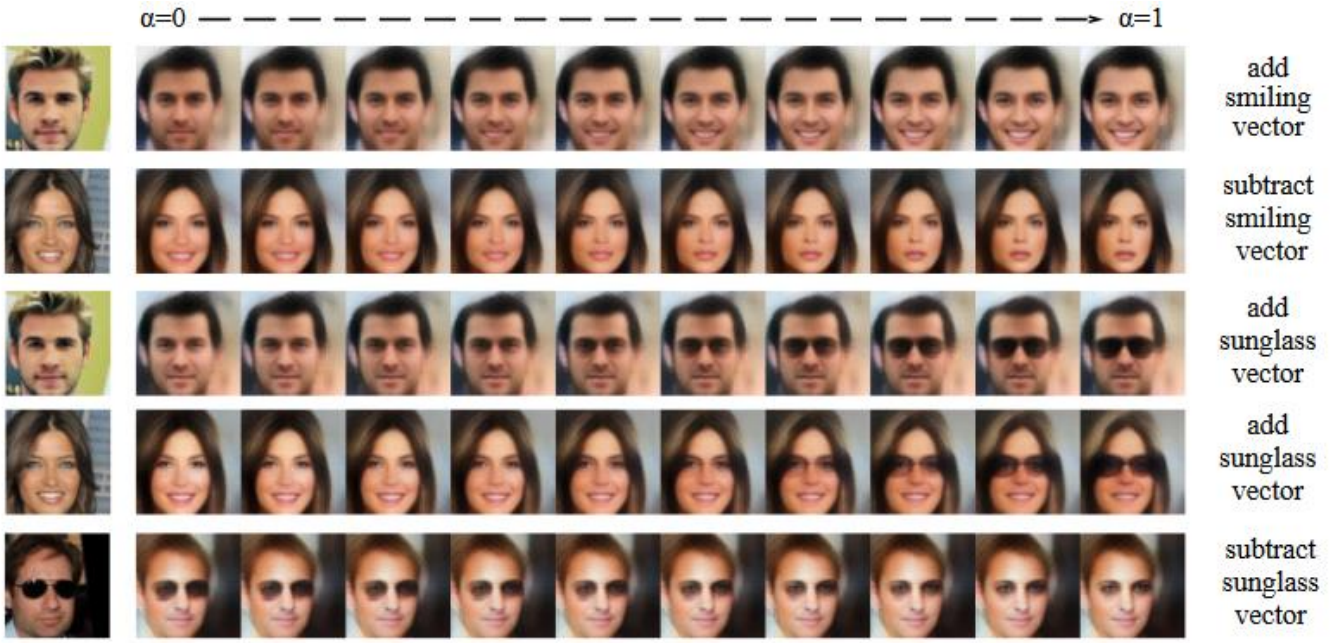
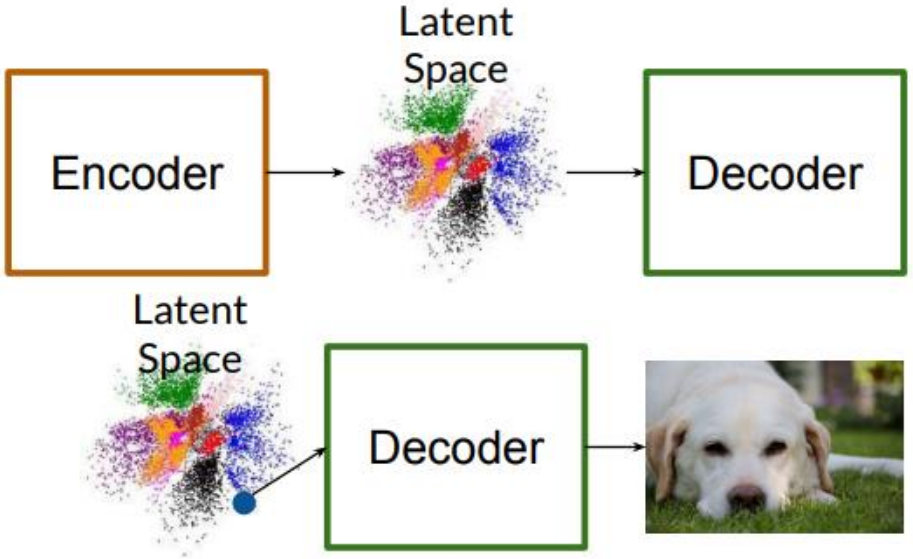


Figure 6. Vector arithmetic for visual attributes. Each row is the generated faces from latent vector z_{left} by adding or subtracting an attribute-specific vector, i.e., $z_{left} + \alpha\ z_{smiling}$, where $\alpha = 0, 0.1, \ldots, 1$. The first row is the transition by adding a smiling vector with a linear factor α from left to right, the second row is the transition by subtracting a smiling vector, the third and fourth row are the results by adding a eyeglass vector to the latent representation for a man and women, and the last row shows results by subtracting an eyeglass vector.

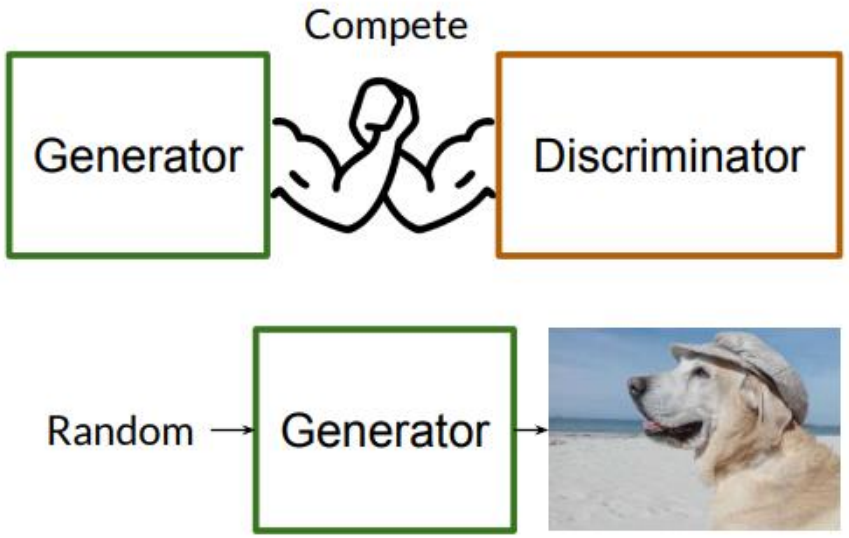
https://arxiv.org/pdf/1610.00291.pdf

Generative Adversarial Networks (GAN)

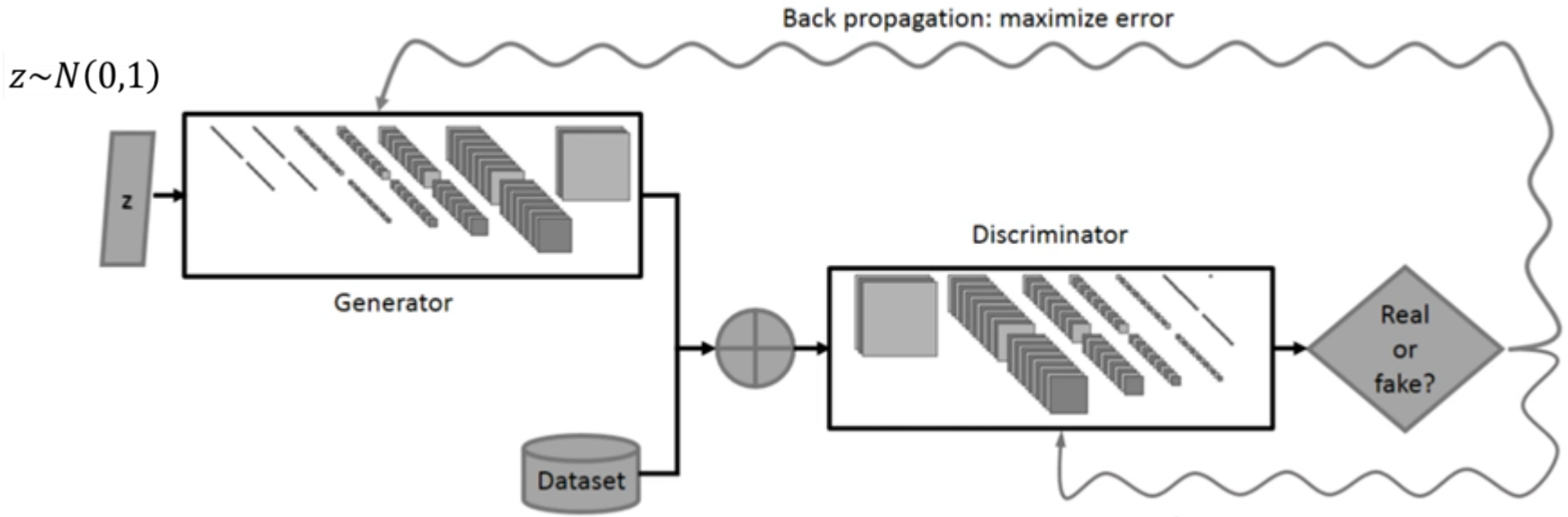
Variational Autoencoders



Generative Adversarial Networks



https://www.[illegible].org/learn/build-basic-generative-adversarial-networks-gans/home/info



for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

https://dlcourse.ai/

https://arxiv.org/pdf/1406.2661.pdf

Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

https://arxiv.org/pdf/1406.2661.pdf

"The GAN Zoo"

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

Ian Goodfellow
@goodfellow_ian

4.5 years of GAN progress on face generation.
arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948

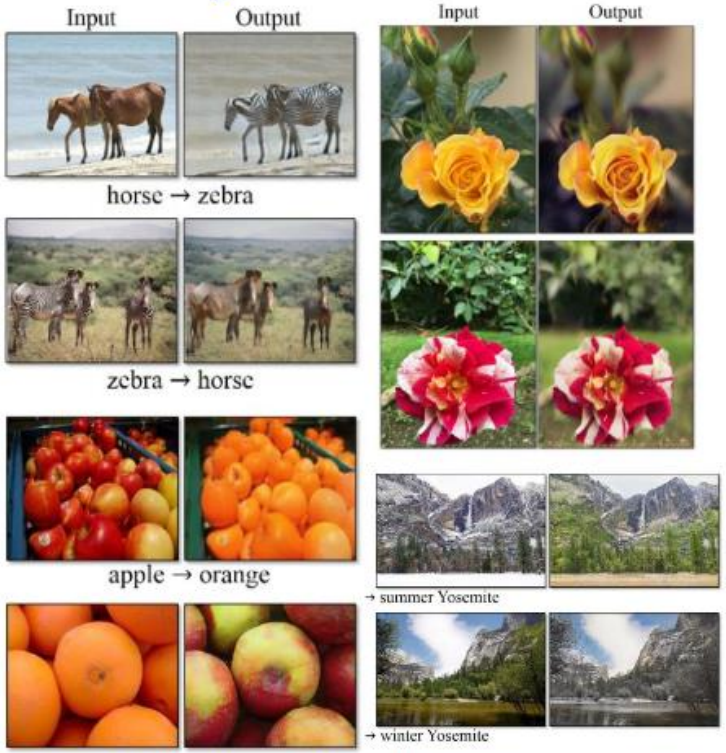
2017: Year of the GAN

Better training and generation

LSGAN. Mao et al. 2017.

BEGAN. Bertholet et al. 2017.

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

Reed et al. 2017.

Many GAN applications

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

https://www.youtube.com/watch?v=[illegible]WoItGTWV54&index=14&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&t=0s

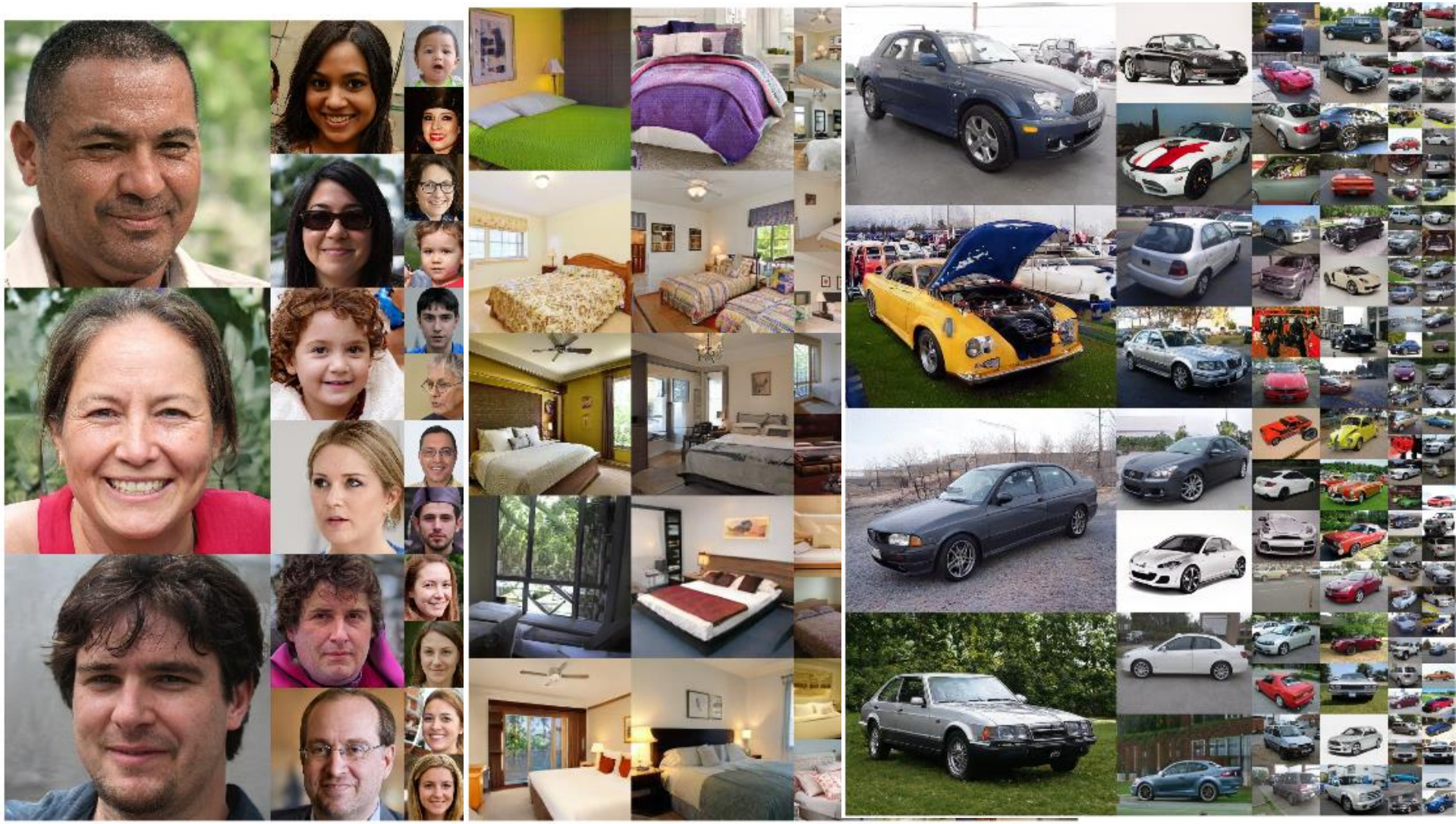
Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

Figure 6: Visual quality comparison in LSUN BEDROOM; pictures copied from the cited articles.

https://arxiv.org/pdf/1710.10196.pdf

Samples from BigGAN: https://openreview.net/pdf?id=B1xsqj09Fm

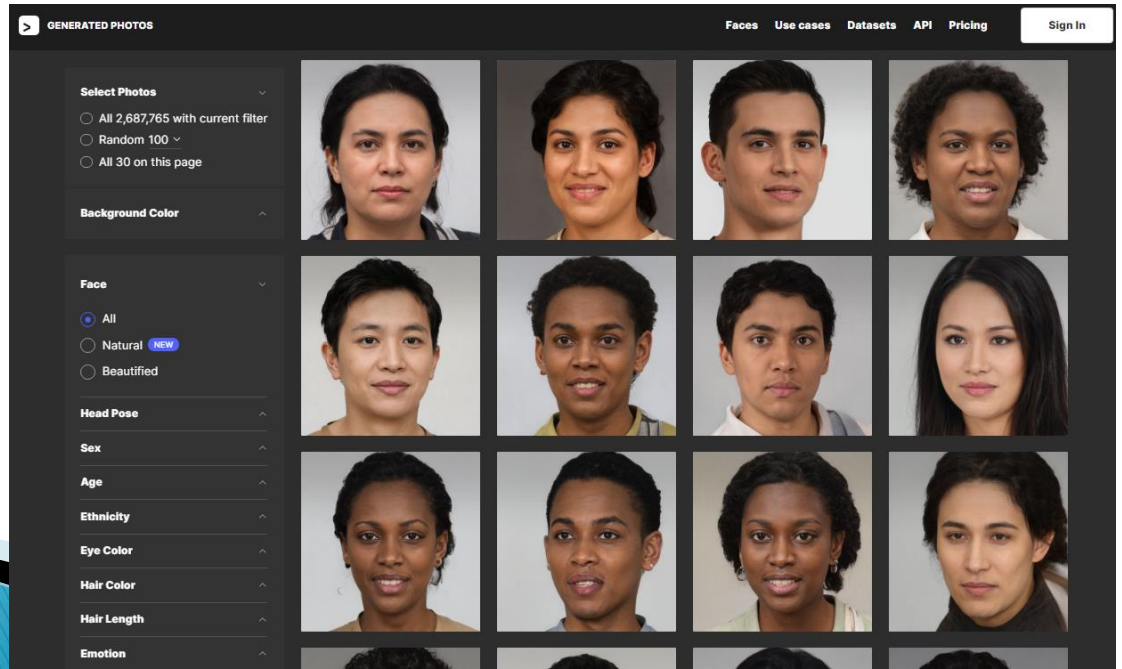
StyleGAN'18



https://arxiv.org/pdf/1812.04948.pdf

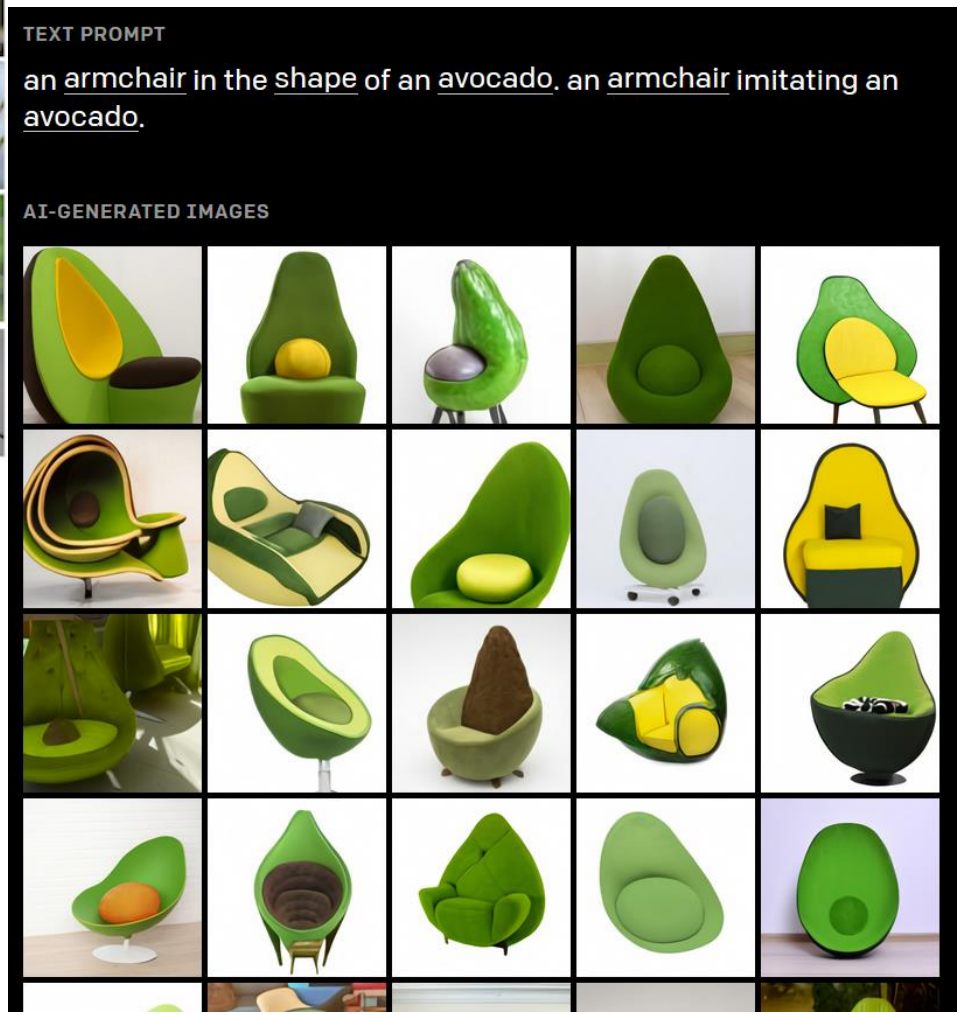
https://thispersondoesnotexist.com/

https://generated.photos/faces

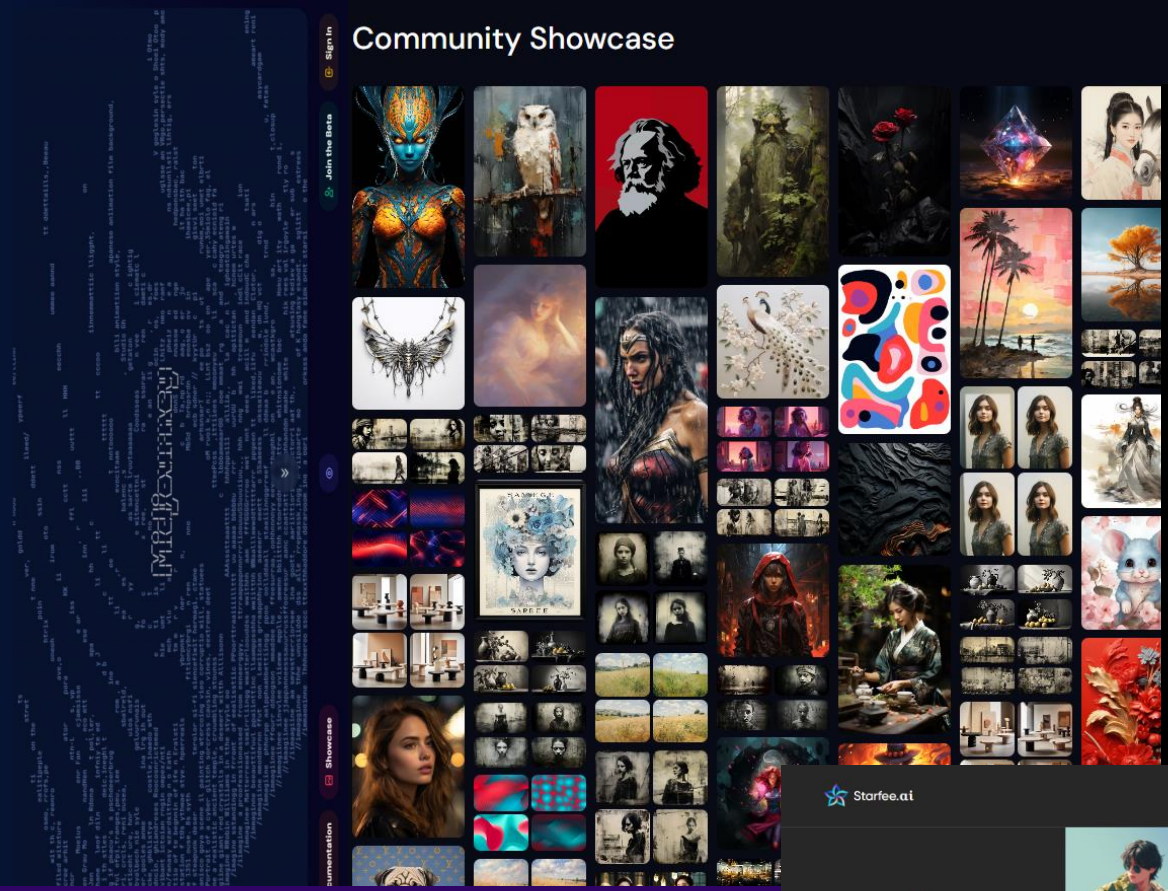


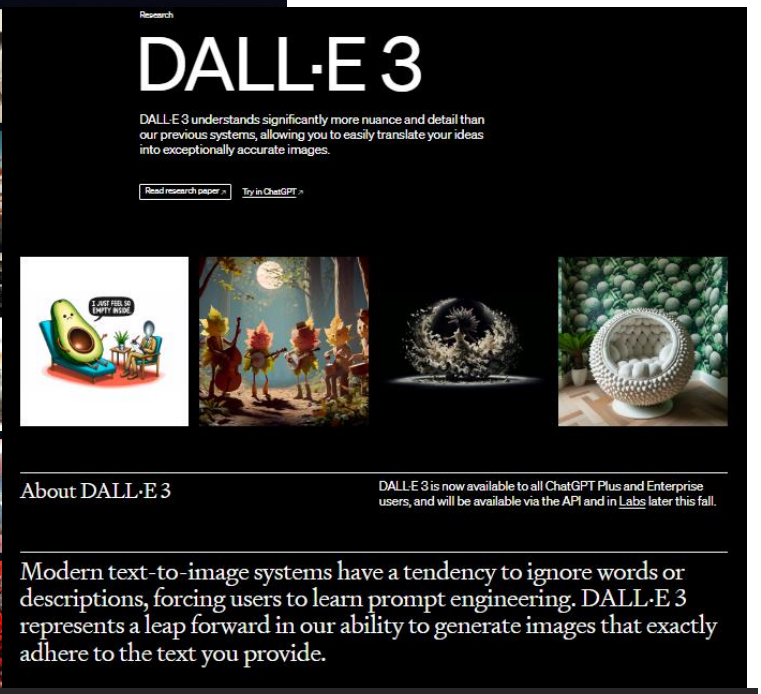
Text-to-Image Generation

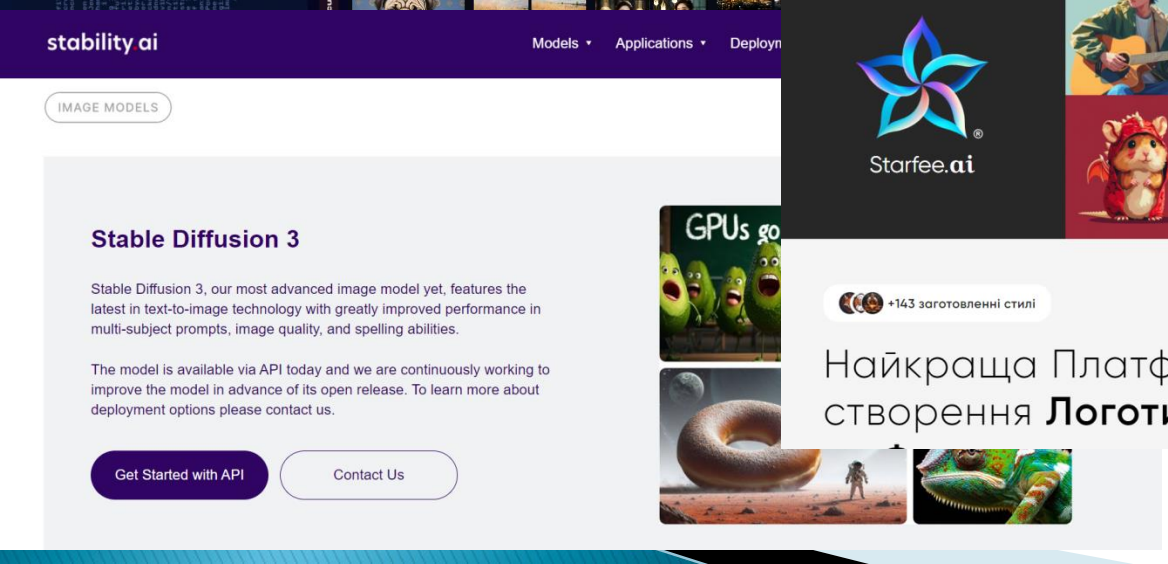
https://paperswithcode.com/task/text-to-image-generation

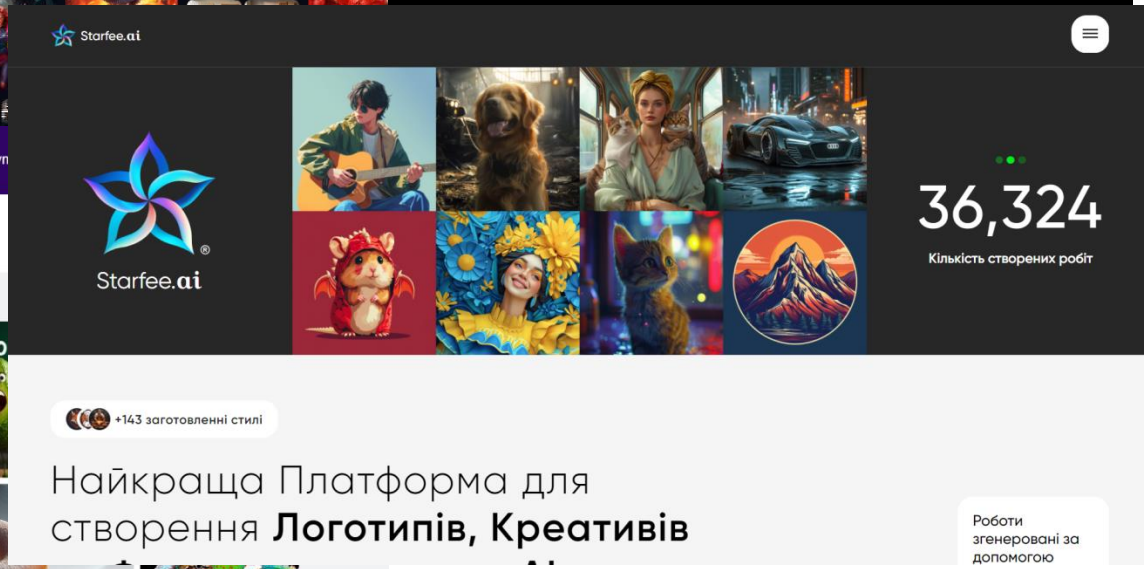


https://openai.com/blog/dall-e/


Community Showcase
Recent
Top
Sign In
Join the Beta
Showcase
Documentation


Research
DALL·E 3
DALL·E 3 understands significantly more nuance and detail than our previous systems, allowing you to easily translate your ideas into exceptionally accurate images.
Read research paper
Try in ChatGPT
About DALL·E 3
DALL·E 3 is now available to all ChatGPT Plus and Enterprise users, and will be available via the API and in Labs later this fall.
Modern text-to-image systems have a tendency to ignore words or descriptions, forcing users to learn prompt engineering. DALL·E 3 represents a leap forward in our ability to generate images that exactly adhere to the text you provide.


stability.ai
Models
Applications
IMAGE MODELS
Stable Diffusion 3
Stable Diffusion 3, our most advanced image model yet, features the latest in text-to-image technology with greatly improved performance in multi-subject prompts, image quality, and spelling abilities.
The model is available via API today and we are continuously working to improve the model in advance of its open release. To learn more about deployment options please contact us.
Get Started with API
Contact Us
GPUs go


Starfee.ai
Starfee.ai
36,324
Кількість створених робіт
+143 заготовленні стилі
Найкраща Платформа для створення Логотипів, Креативів
Роботи згенеровані за допомогою

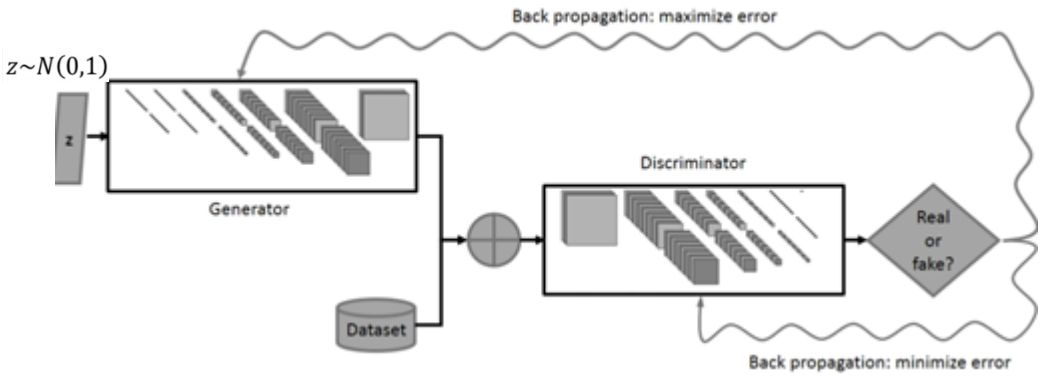
GANs for 3D Objects

3D-GAN

Generative Design

Wu, Jiajun, et al. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." *Advances in neural information processing systems*. 2016.

https://www.[illegible].org/learn/build-basic-generative-adversarial-networks-gans/home/info



Unconditional

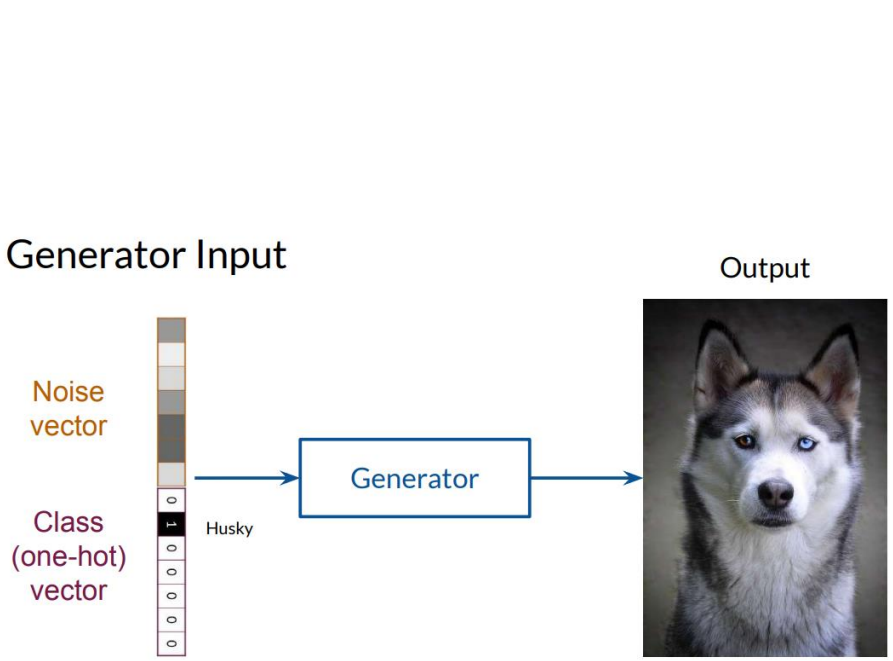
Examples from *random classes*

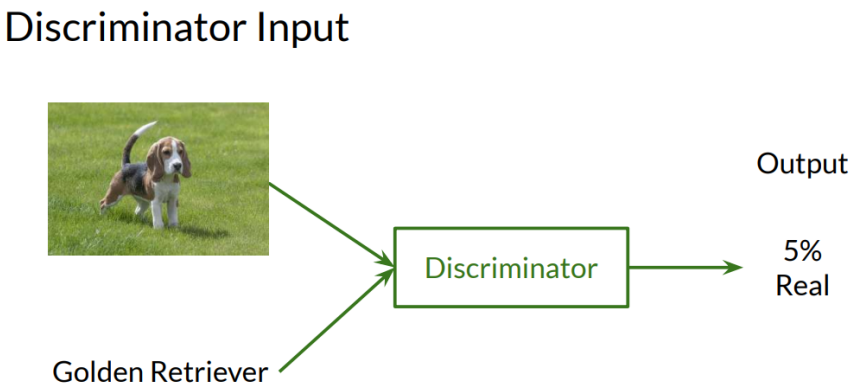
Training dataset *doesn't need to be labeled*

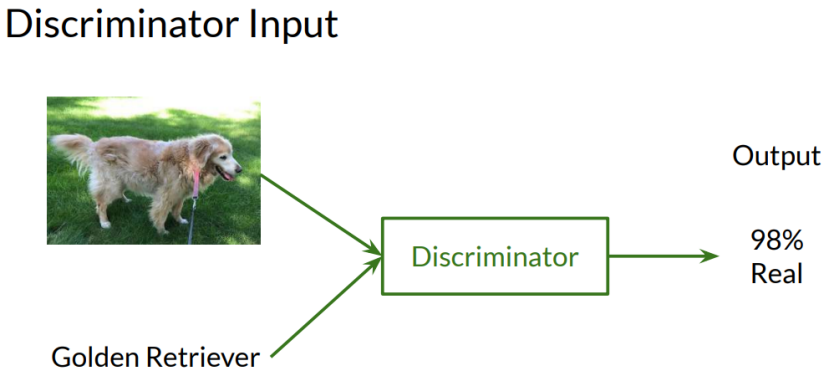
Controllable	Conditional
Examples with the **features that you want**	Examples from *the classes you want*
Training dataset **doesn't need to be labeled**	Training dataset *needs to be labeled*
Manipulate the z vector input	*Append a class vector* to the input

https://www.[illegible].org/learn/build-basic-generative-adversarial-networks-gans/home/info

Conditional generation







https://www.[illegible].org/learn/build-basic-generative-adversarial-networks-gans/home/info

Controllable generation

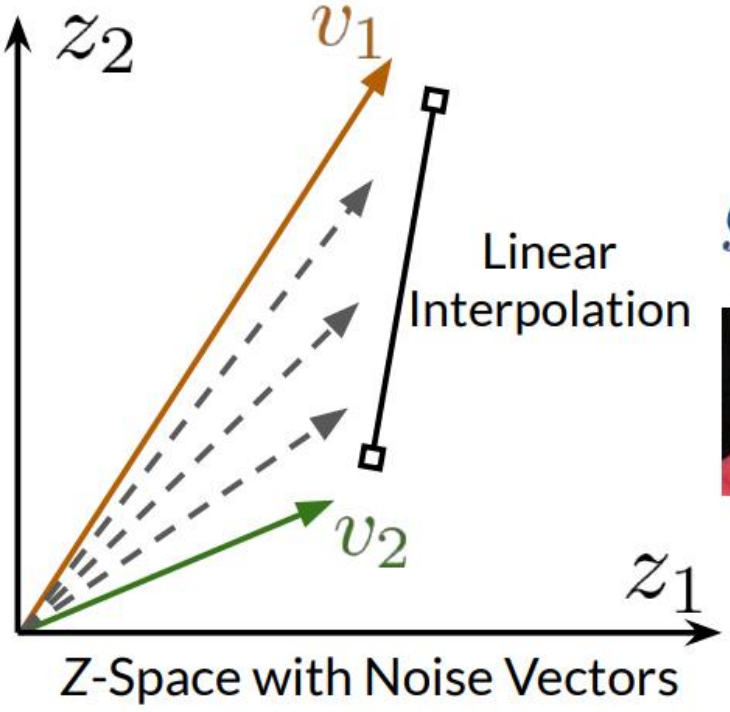
Change specific features of the output

Tweak the input noise vector to get different features on the output

1907.10786.pdf (arxiv.org)

https://www.[illegible].org/learn/build-basic-generative-adversarial-networks-gans/home/info

Interpolation Using the Z-Space



Z-Space with Noise Vectors

Intermediate images using
the Z-space

$g(v_1)$ ⟷ $g(v_2)$

https://www.[illegible].org/learn/build-basic-generative-adversarial-networks-gans/home/info

Adversarial attack

$\boldsymbol{x}$

"panda"
57.7% confidence

$+.007 \times$

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

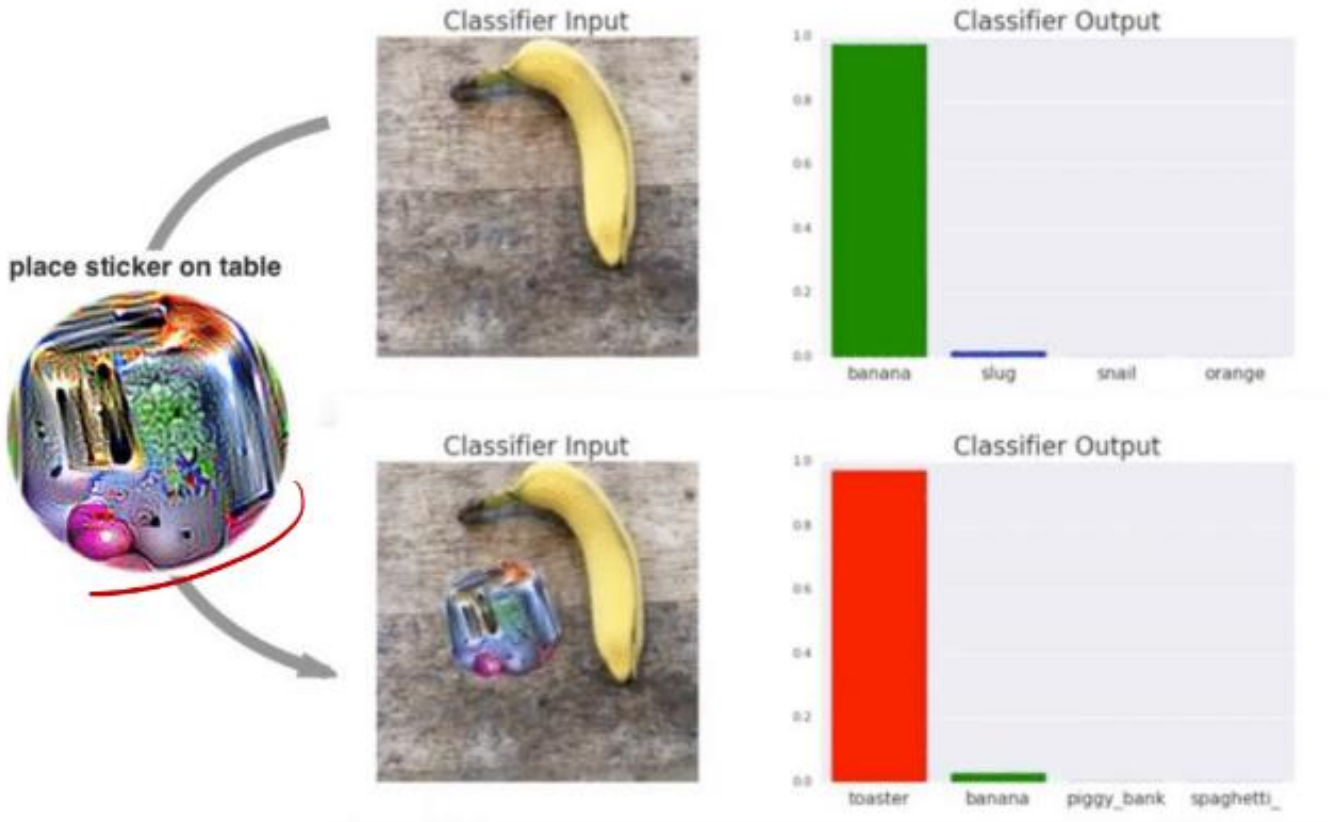
$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

http://karpathy.github.io/2015/03/30/breaking-convnets/

Adversarial Patch



https://arxiv.org/abs/1712.09665

Convolutional Neural Networks

by DeepLearning.AI

Convolutional Neural Networks
DeepLearning.AI

Матеріал курсу

- Тиждень 1
- Тиждень 2
- Тиждень 3
- Тиждень 4

Оцінки

Примітки

Повідомлення

Ресурси

курс: загальні відомості
In the fourth course of the Deep Learning Specialization, you will understand how computer vision has evolved and become familiar with its exciting applications such as autonomous driving, face recognition, reading radiology images, and more.

By the end, you will be able to build a convolutional neural network, including recent variations such as residual networks; apply convolutional networks to visual detection and recognition tasks; and use neural style transfer to generate art and apply these algorithms to a variety of image, video, and other 2D or 3D data.

The Deep Learning Specialization is our foundational program that will help you understand the capabilities, challenges, and consequences of deep learning and prepare you to participate in the development of leading-edge AI technology. It provides a pathway for you to gain the knowledge and skills to apply machine learning to your work, level up your technical career, and take the definitive step in the world of AI.

Показати менше

Викладають: **Andrew Ng**, Instructor
Founder, DeepLearning.AI & Co-founder, Coursera

https://www.coursera.org/learn/convolutional-neural-networks/home/info

Advanced Computer Vision with TensorFlow

by DeepLearning.AI

Advanced Computer Vision with TensorFlow
DeepLearning.AI

- **Матеріал курсу**
  - Тиждень 1
  - Тиждень 2
  - Тиждень 3
  - Тиждень 4
- **Оцінки**
- **Примітки**
- **Повідомлення**
- **Інформація**

курс: загальні відомості
In this course, you will:

a) Explore image classification, image segmentation, object localization, and object detection. Apply transfer learning to object localization and detection.
b) Apply object detection models such as regional-CNN and ResNet-50, customize existing models, and build your own models to detect, localize, and label your own rubber duck images.
c) Implement image segmentation using variations of the fully convolutional network (FCN) including U-Net and d) Mask-RCNN to identify and detect numbers, pets, zombies, and more.
d) Identify which parts of an image are being used by your model to make its predictions using class activation maps and saliency maps and apply these ML interpretation methods to inspect and improve the design of a famous network, AlexNet.

The DeepLearning.AI TensorFlow: Advanced Techniques Specialization introduces the features of TensorFlow that provide learners with more control over their model architecture and tools that help them create and train advanced ML models.

This Specialization is for early and mid-career software and machine learning engineers with a foundational understanding of TensorFlow who are looking to expand their knowledge and skill set by learning advanced TensorFlow features to build powerful models.

Показати менше

Викладають: **Laurence Moroney**, Instructor
Lead AI Advocate, Google

https://[redacted]coursera.org/learn/advanced-computer-vision-with-tensorflow/home/info

Generative Deep Learning with TensorFlow

by DeepLearning.AI

курс: загальні відомості
In this course, you will:
a) Learn neural style transfer using transfer learning: extract the content of an image (eg. swan), and the style of a painting (eg. cubist or impressionist), and combine the content and style into a new image.
b) Build simple AutoEncoders on the familiar MNIST dataset, and more complex deep and convolutional architectures on the Fashion MNIST dataset, understand the difference in results of the DNN and CNN AutoEncoder models, identify ways to de-noise noisy images, and build a CNN AutoEncoder using TensorFlow to output a clean image from a noisy one.
c) Explore Variational AutoEncoders (VAEs) to generate entirely new data, and generate anime faces to compare them against reference images.
d) Learn about GANs; their invention, properties, architecture, and how they vary from VAEs, understand the function of the generator and the discriminator within the model, the concept of 2 training phases and the role of introduced noise, and build your own GAN that can generate faces.

The DeepLearning.AI TensorFlow: Advanced Techniques Specialization introduces the features of TensorFlow that provide learners with more control over their model architecture, and gives them the tools to create and train advanced ML models.

This Specialization is for early and mid-career software and machine learning engineers with a foundational understanding of TensorFlow who are looking to expand their knowledge and skill set by learning advanced TensorFlow features to build powerful models.

Показати менше

Викладають: Laurence Moroney, Instructor
Lead AI Advocate, Google

Examples

- Segmentation

  https://colab.research.google.com/github/https-deeplearning-ai/tensorflow-3-public/blob/main/Course%203%20-%20Advance%20Computer%20Vision/W3/ungraded_labs/C3_W3_Lab_2_OxfordPets-UNet.ipynb#scrollTo=_xzI28AfxFQi

- Object_Localization

- https://colab.research.google.com/github/https-deeplearning-ai/tensorflow-3-public/blob/main/Course%203%20-%20Advance%20Computer%20Vision/W1/ungraded_labs/C3_W1_Lab_3_Object_Localization.ipynb

- Object Detection with TF Hub

- https://colab.research.google.com/github/https-deeplearning-ai/tensorflow-3-public/blob/main/Course%203%20-%20Advance%20Computer%20Vision/W2/ungraded_labs/C3_W2_Lab_1_Simple_Object_Detection.ipynb
- https://colab.research.google.com/github/https-deeplearning-ai/tensorflow-3-public/blob/main/Course%203%20-%20Advance%20Computer%20Vision/W2/ungraded_labs/C3_W2_Lab_2_Object_Detection.ipynb#scrollTo=CxmDMK4yupqg
- https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb
- https://colab.research.google.com/github/tensorflow/models/blob/master/research/object_detection/colab_tutorials/eager_few_shot_od_training_tf2_colab.ipynb

First GAN

- https://colab.research.google.com/github/https-deeplearning-ai/tensorflow-3-public/blob/main/Course%204%20-%20Generative%20Deep%20Learning/W4/ungraded_labs/C4_W4_Lab_1_First_GAN.ipynb