Neural Networks

Nasim Rahaman & Nicolas Roth

Fundamentals

Good Old Fashioned (Symbolic) AI

Focused on **logical reasoning** instead of semantic cues.

The AI, in itself, is a bunch of **rules** which it uses to arrive at a **conclusion** given a set of **predicates**. But the system knows nothing about the **semantics** of the rules/predicates.

In the following, we consider the scenario where Carol works_at a restaurant as a waitress, Alice orders a pizza.

Predicates would be:
works_at(restaurant,waitress, Carol)
orders(Alice, pizza)

A rule could say:
if works_at(restaurant,waitress, A) && orders(B, food) → then serves(A, food, B)

Conclusion: Carol serves pizza to Alice.
serves(Carol,pizza,Alice)

But what does it mean to order a pizza?

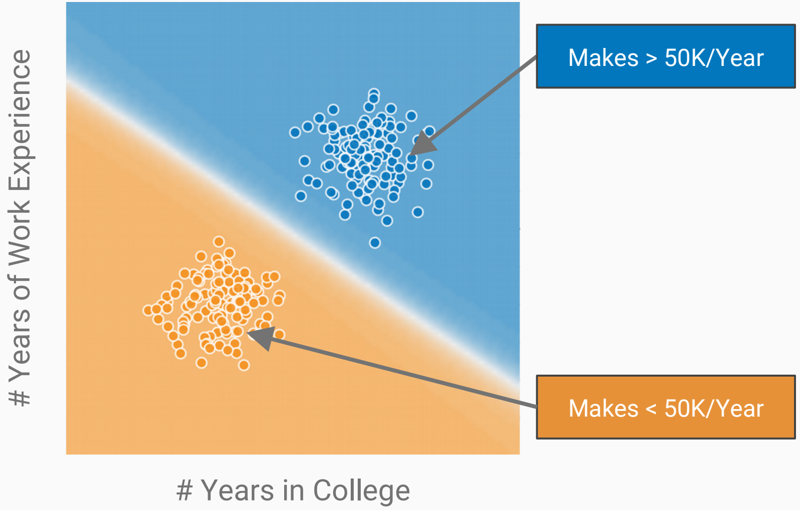
Symbolic AI is not exactly very popular these days...

Machine Learning: A Framework for Data-Driven AI

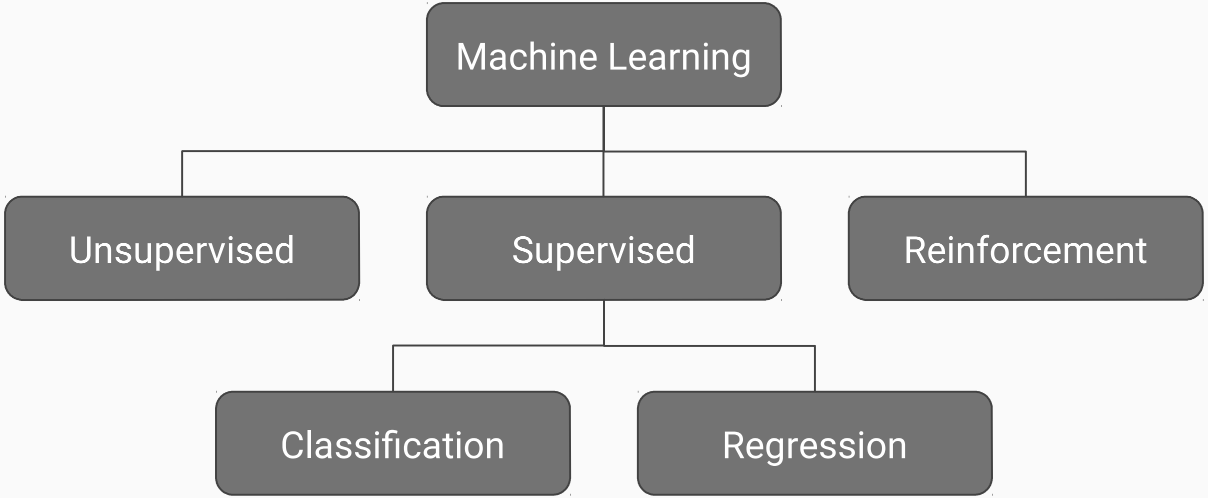
The Big Idea: **Learn Models from Data**.

Example Problem: Given the number of years spent in college and work experience, predict if a person makes more or less than $50K/year.

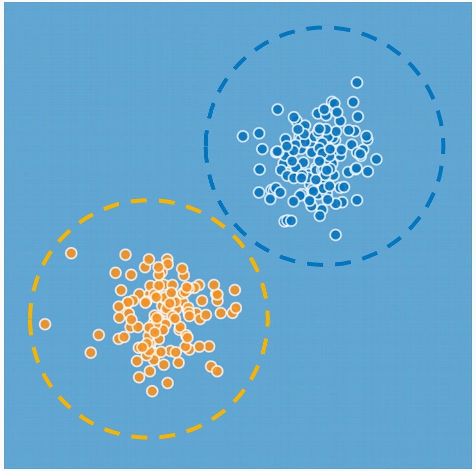
Given: Some training data obtained by polling (say) 100 individuals whether they make more than 50K a year and the time they've spent in college and at work.



Taxonomy of Machine Learning Algorithms

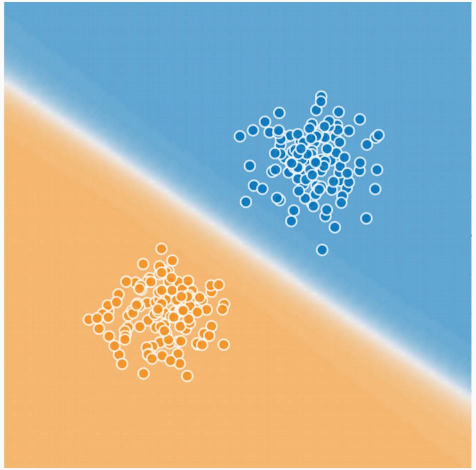


Supervised vs. Unsupervised Learning



Unsupervised Learning

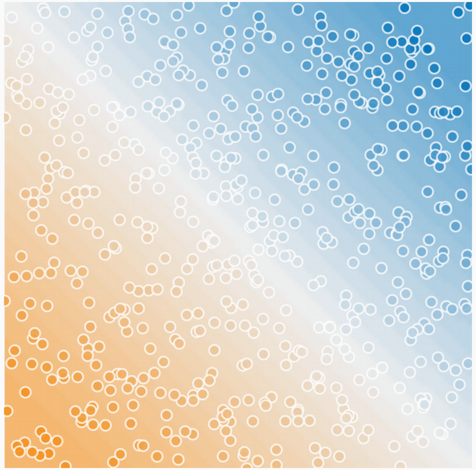
The *label* (i.e. the knowledge if the person makes more than 50K a year) is not known, but we could still **find patterns** in the data, e.g. by clustering.



Supervised Learning

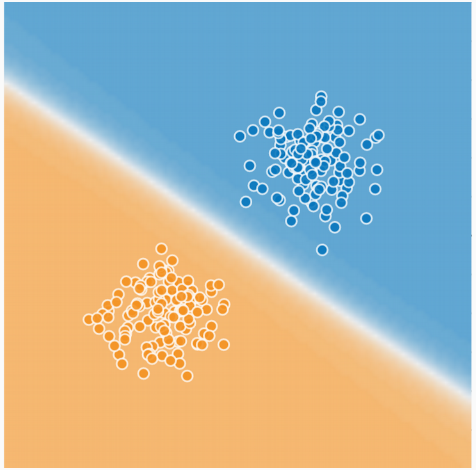
The *label* is given, and we require the model to **predict** a label given new input.

Classification vs. Regression Problems



Regression

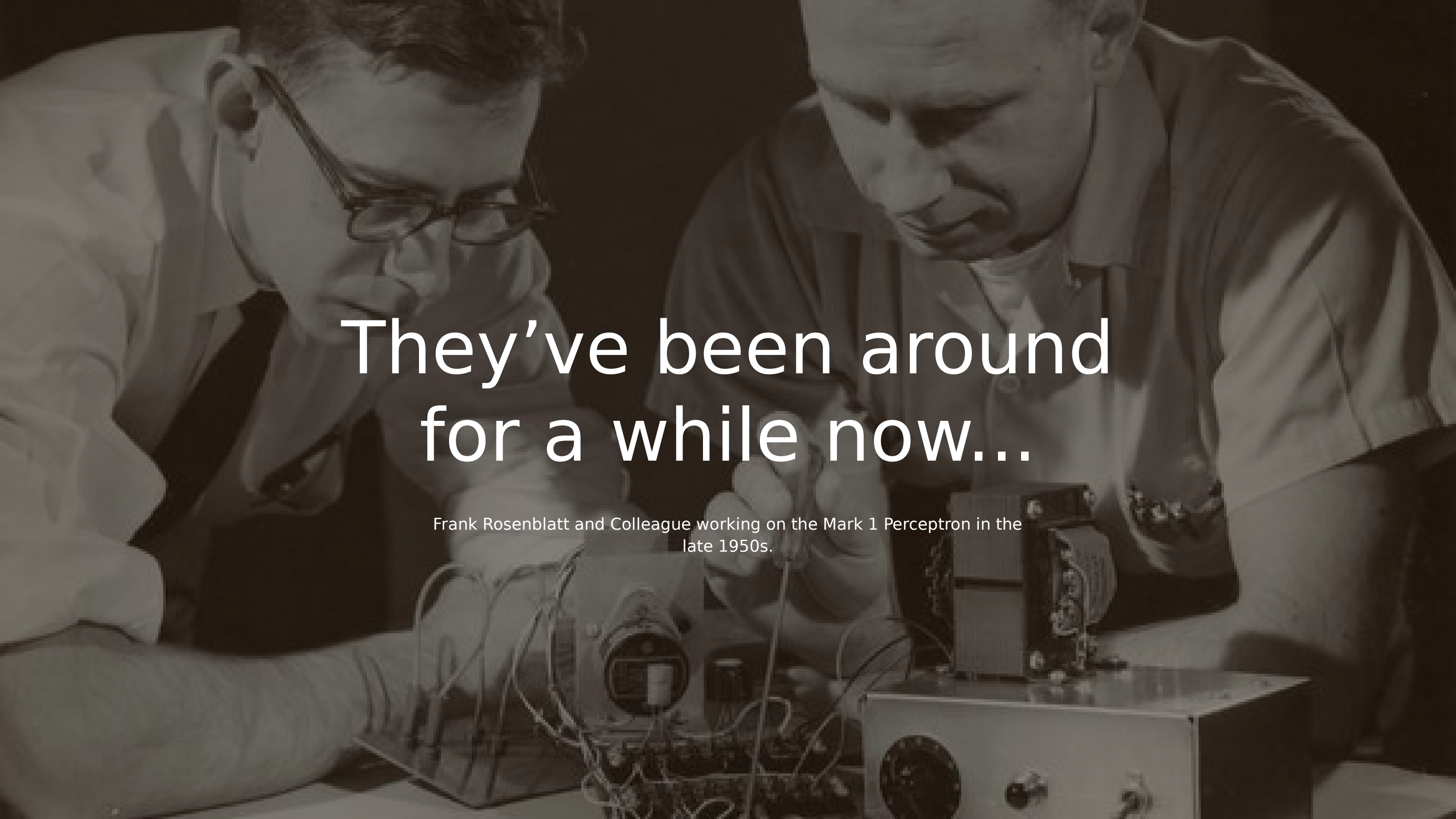
The label is a continuous number, specifying **exactly how much** the person makes.



Classification

The label is categorical, i.e. it says **whether** a person makes more or less than 50K.

Neural Networks



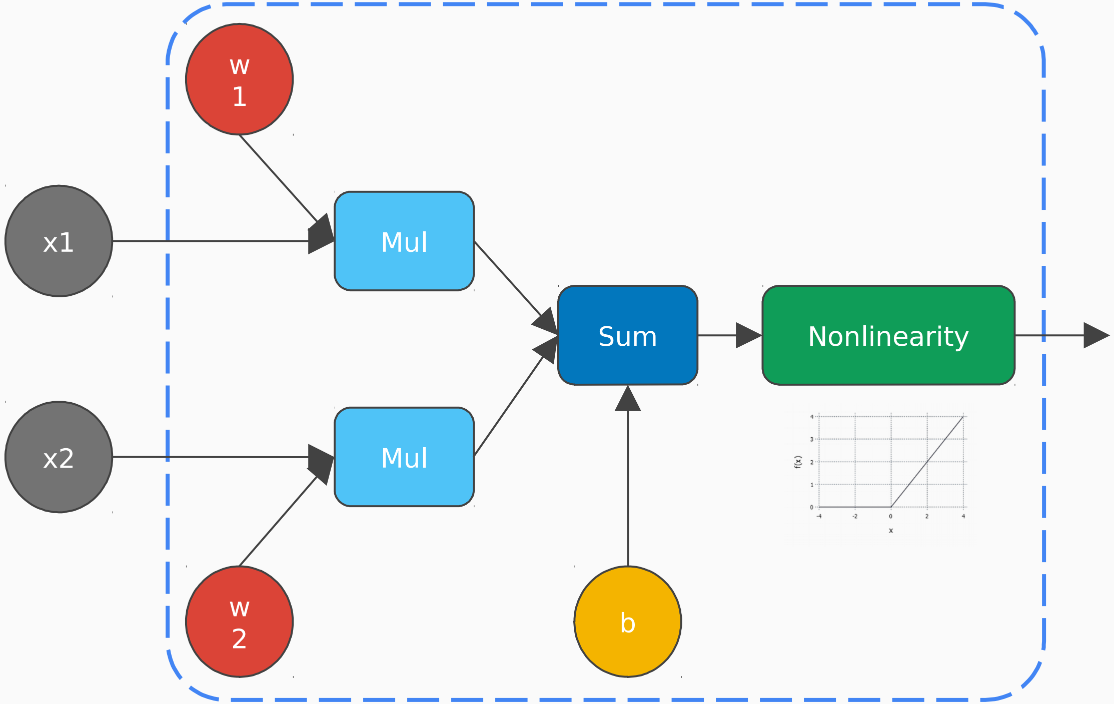
They’ve been around for a while now…

Frank Rosenblatt and Colleague working on the Mark 1 Perceptron in the late 1950s.

The Perceptron as the Building Blocks of Neural Networks

The perceptron is the simplest possible neural network, also often called a **neuron**.

Mathematically, it can be expressed as a scalar product between two vectors, followed by an application of some nonlinear function.

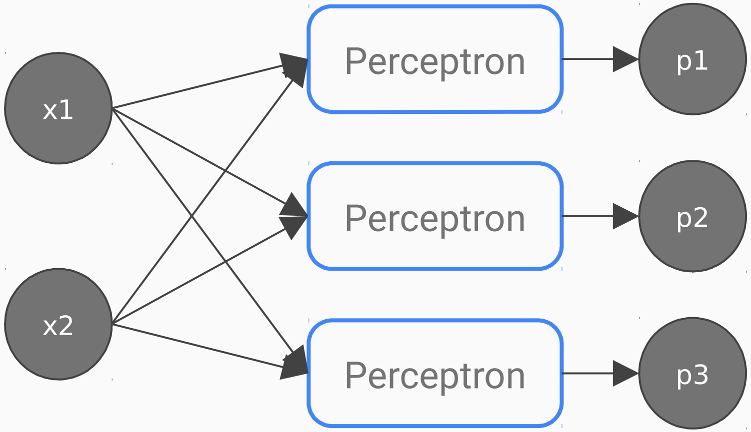


$$p(x_1, x_2) = \sigma\left(\begin{pmatrix} w_1 & w_2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + b\right)$$

Stacking Perceptrons to make a Layer

A **layer** in a neural network is a stack of perceptrons.

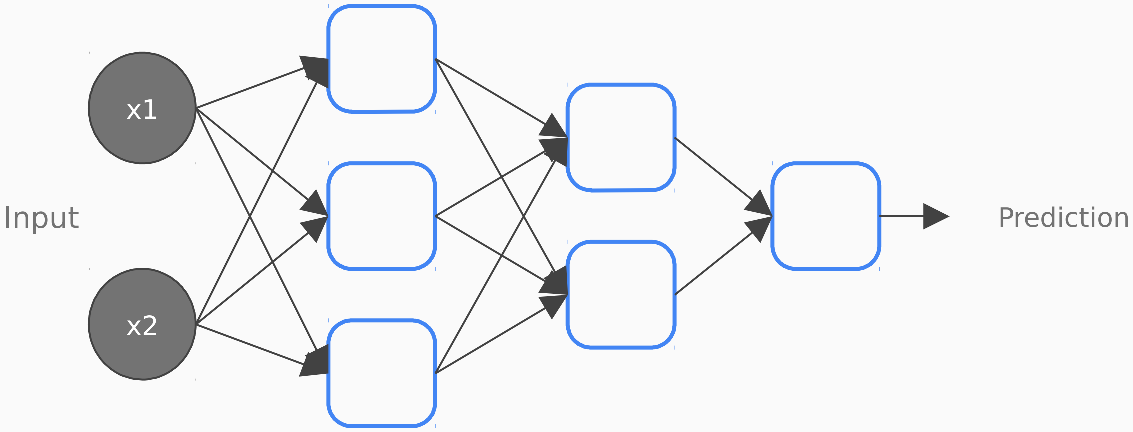
Mathematically, it can be expressed as a matrix multiplication followed by the elementwise application of a nonlinear function.



$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \sigma\left(\begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}\right)$$

Composing Layers to make a MLP

Multiple layers can be composed to make a **multi-layered perceptron**, or simply a **fully-connected deep neural network**.



A multi-layer perceptron of 3 layers with (3, 2, 1) neurons.

Automatic Differentiation

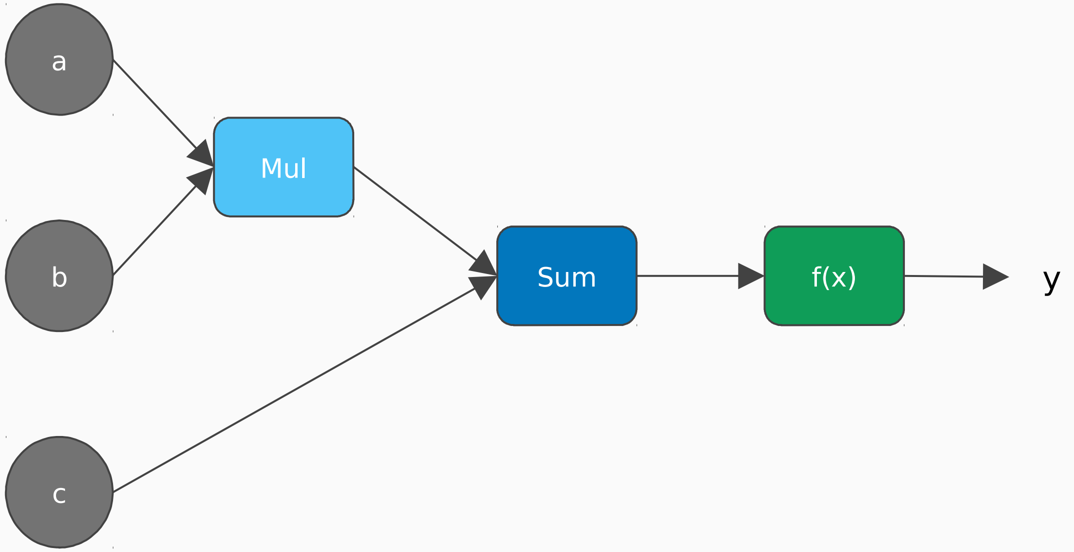
Make Networks Train Again!

Production neural networks often have tens of millions of weights and bias parameters. Automatic Differentiation (or **backpropagation**) is crucial for training.

Computational Graph

A computational graph is a graphical way of representing a mathematical expression.

Why do we care? Neural networks can be expressed as computational graphs.



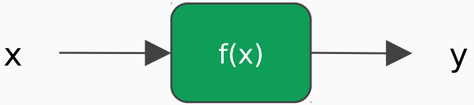
$$y = f((a * b) + c)$$

Forward & Backward Pass

A forward pass through a node in a computational graph is the same as evaluating it for a given input.

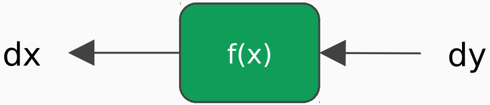
A backward pass through a node means to evaluate the gradient of some **dummy function** with respect to the input of the node from the gradient of the same function with respect to the output of the node.

The gradients of this dummy function w.r.t to a given variable is called the **delta** of the variable.



$$y = f(x)$$

Forward



$$\frac{\partial}{\partial x}(\cdot) = \frac{\partial f}{\partial x}\frac{\partial}{\partial f}(\cdot) = \frac{\partial f}{\partial x}\frac{\partial}{\partial y}(\cdot)$$

Backward

Forward & Backward Pass

A forward pass through a node in a computational graph is the same as **evaluating it for a given input**.

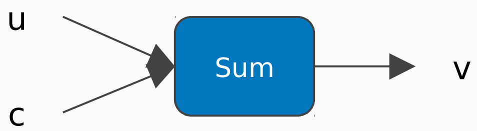
A backward pass, on the other hand, is equivalent to **computing the gradient of the output with respect to the input**.

$$u = ab$$

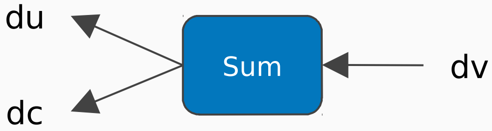
$$\frac{\partial}{\partial a} = \frac{\partial (ab)}{\partial a} \frac{\partial}{\partial u}(\cdot) = b \frac{\partial}{\partial u}(\cdot)$$

Forward

Backward



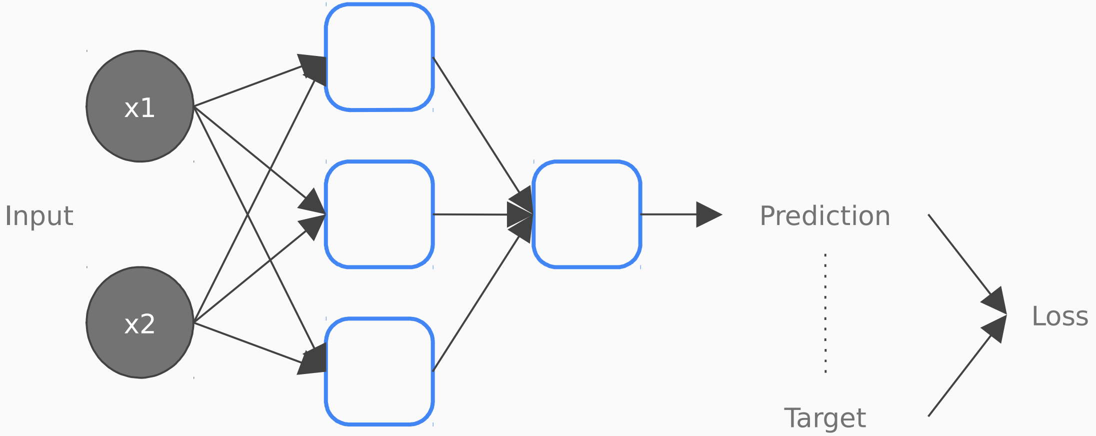
$$v = u + c$$



$$\frac{\partial}{\partial u}(\cdot) = \frac{\partial (u+c)}{\partial u} \frac{\partial}{\partial v}(\cdot) = \frac{\partial}{\partial v}(\cdot)$$

AutoDiff and Neural Networks

Automatic differentiation can be used to compute the gradient of a **loss function** with respect to the parameters of the network.



$$Loss = (1/2) * (Prediction - Target)^2$$

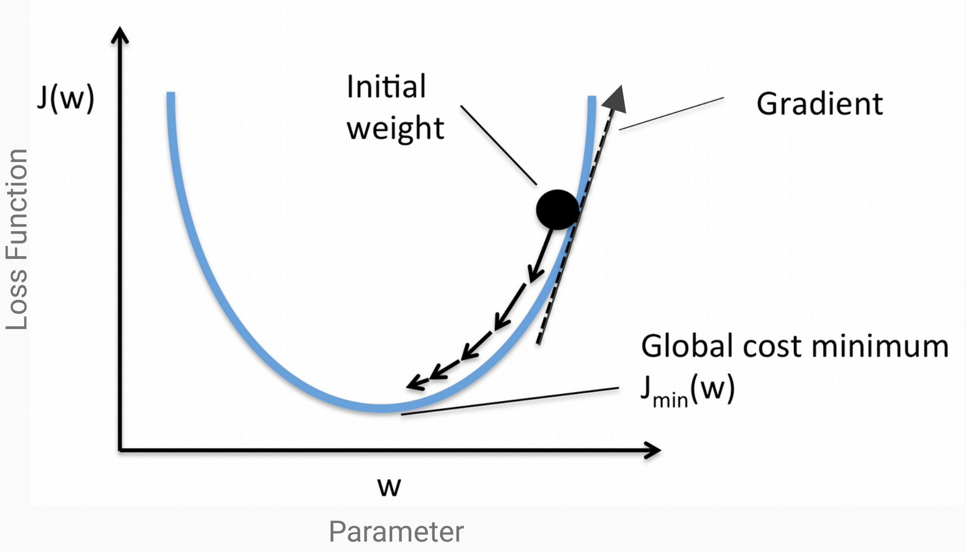
$$d(Prediction) = Prediction - Target$$

$$d(w) = ..., d(b) = ...$$

Optimization with Gradient Descent

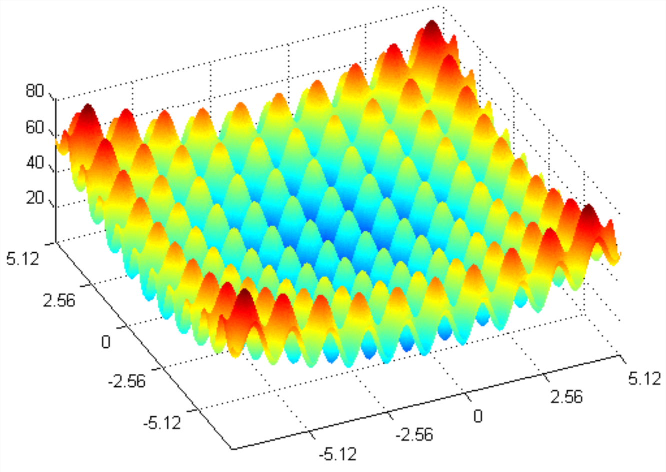
We have the gradients w.r.t the cost function.

We use this gradient to **descent** into a (local) minimum of the loss function.

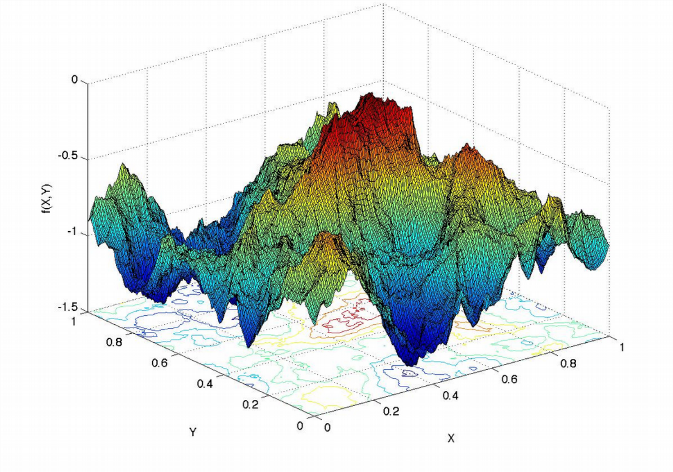


$$w \rightarrow w - learning\_rate * dw$$

But the optimization problem could look like anything between:



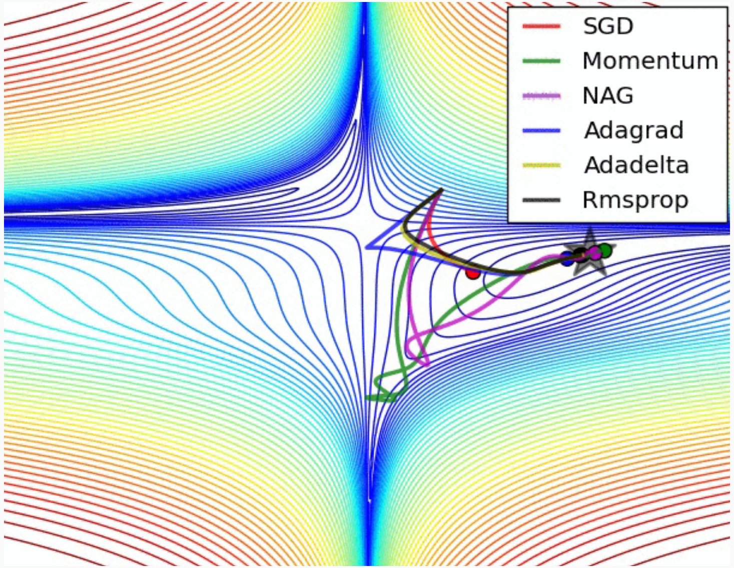
and

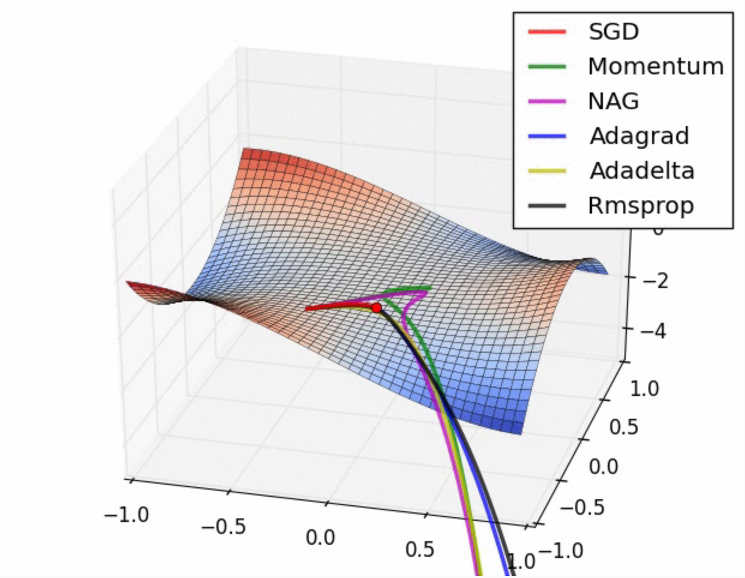


http://descriptor.blogspot.de/2012/11/non-convex-function-rastrigin.html
https://www.cs.bham.ac.uk/research/projects/ecb/displayDataset.php?id=150

The optimization problem is anything but easy.

Sophisticated Gradient-Based Optimizers do tend to help...





Click Me:
http://imgur.com/a/Hqolp

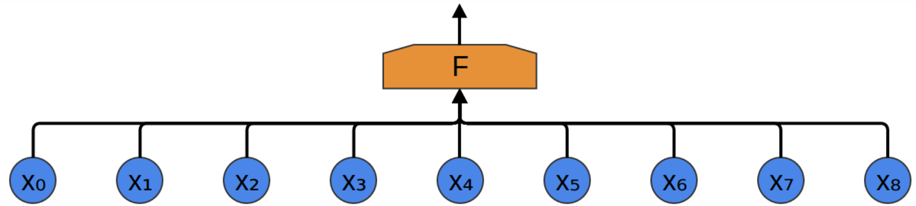
Live Demo

http://playground.tensorflow.org/

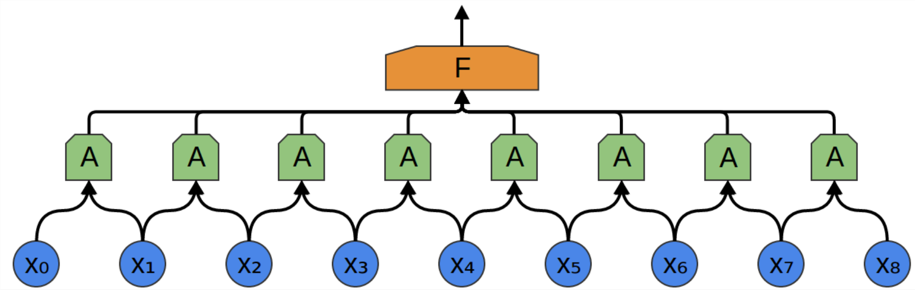
Convolutional Neural Networks

The Idea: Local Connectivity

In fully-connected layers: a perceptron sees **all inputs**.

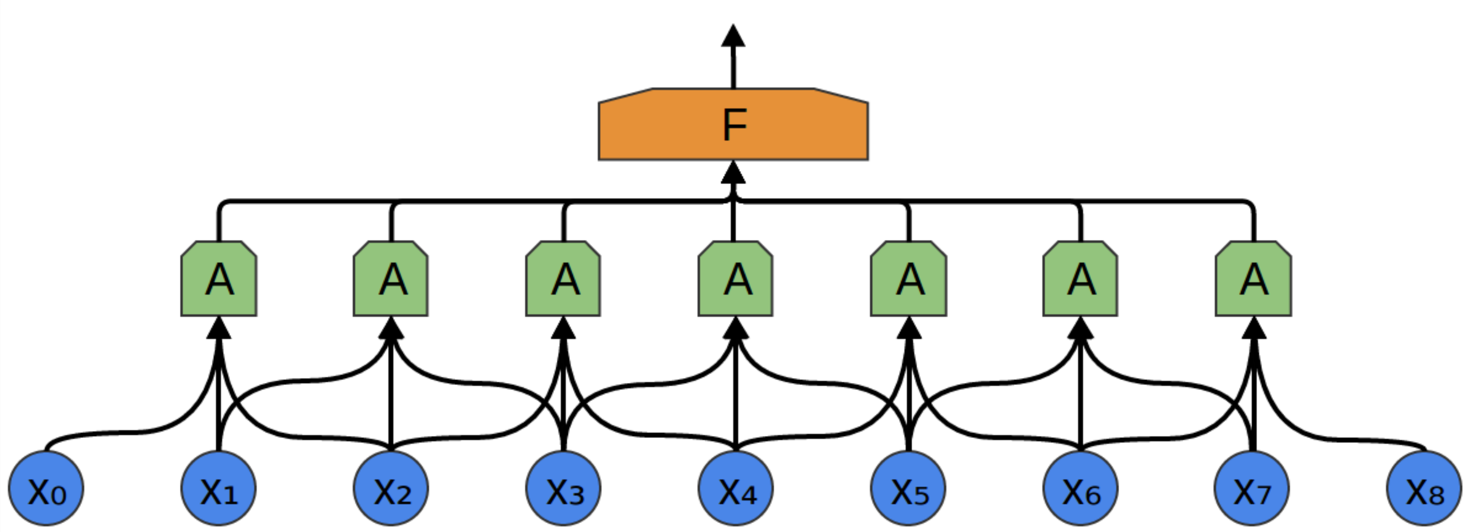


In convolutional layers: perceptrons only see **a neighborhood of the input** at a time.



http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

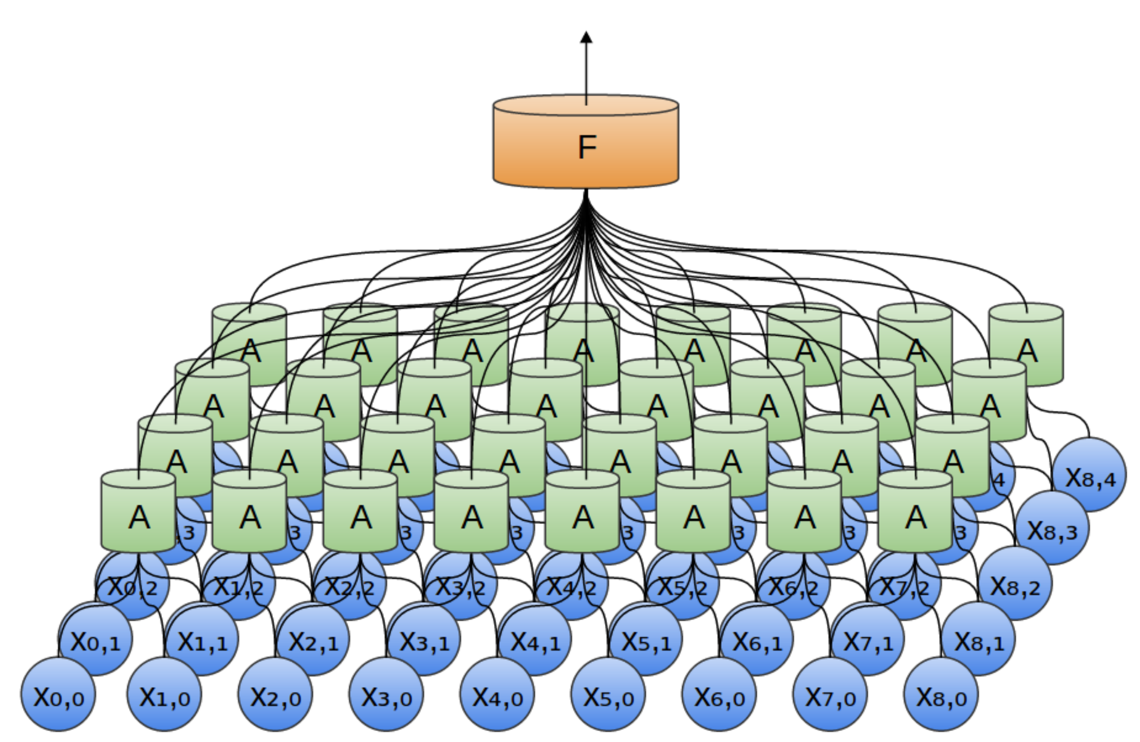
The Idea: Weight Sharing



The perceptron layer **A** is **shared** between all neighborhoods of the input.

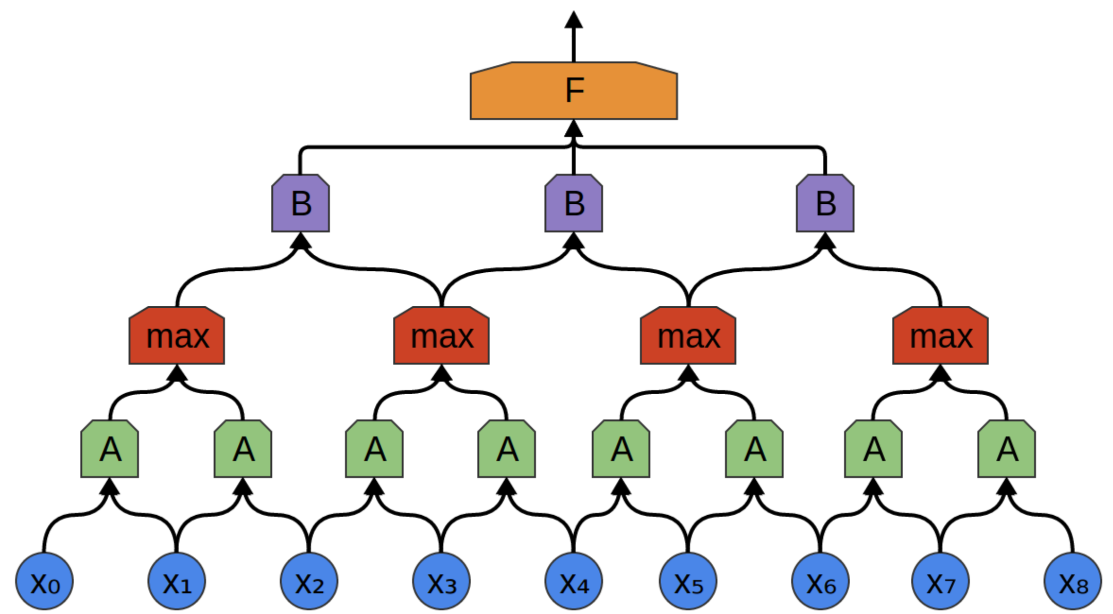
http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

The Idea: in 2D



http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

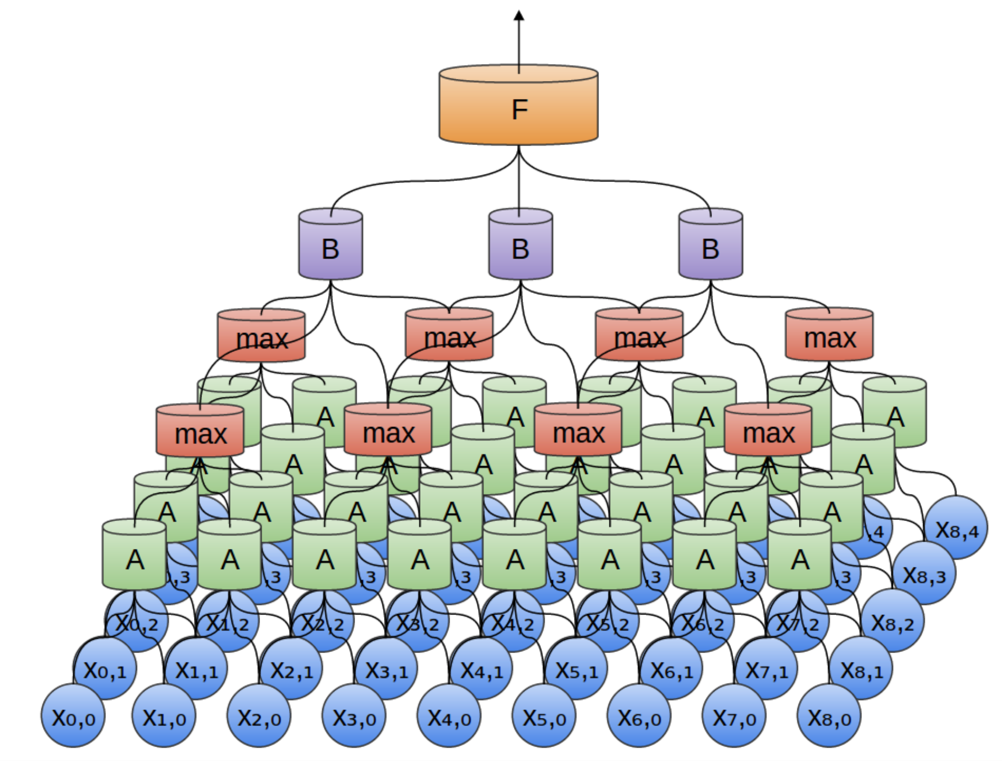
The Idea: Max-Pooling



Max pooling layers are often used to aggregate spatial context.

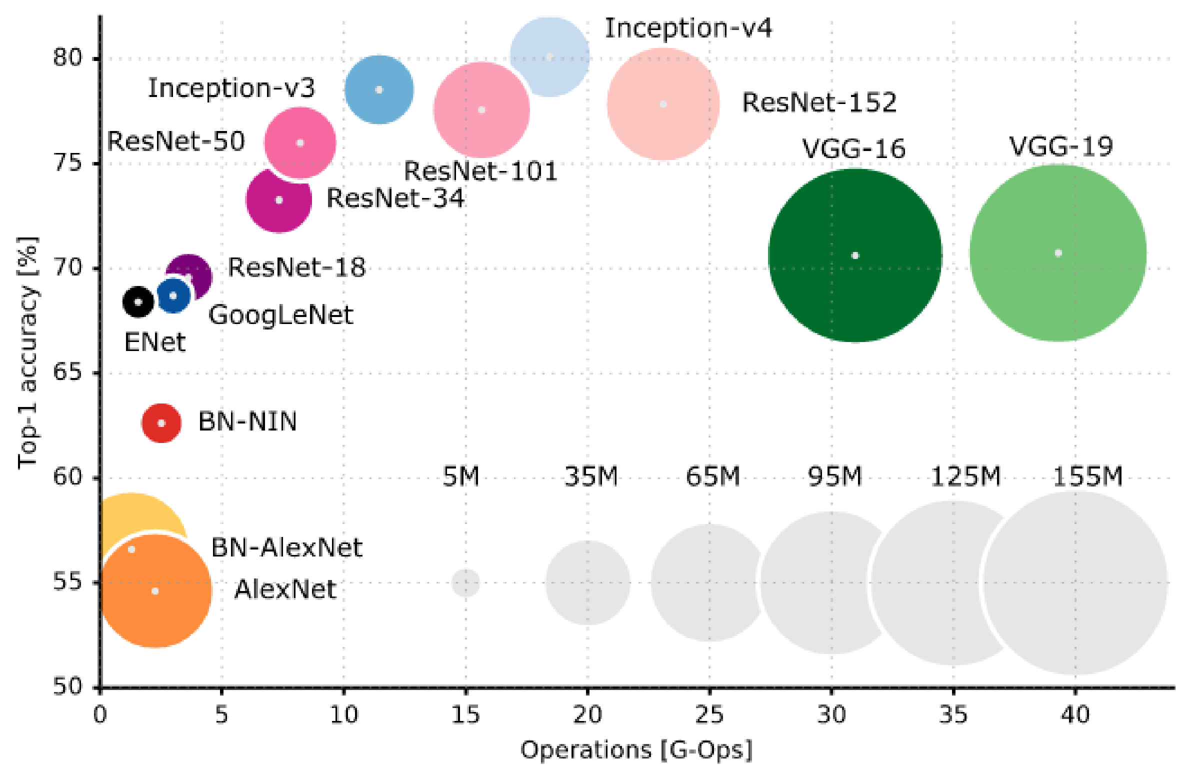
http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

The Idea: Implement with nD Convolutions



http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Popular Network Architectures



https://culurciello.github.io/tech/2016/06/04/nets.html

Live Demo

CS231n: Convolutional Neural Networks for Visual Recognition

Spring 2017

*This network is running live in your browser

Learn more about Convolutional Neural Networks

Conv Nets: A Modular Perspective

Posted on July 8, 2014

neural networks, deep learning, convolutional neural networks, modular neural networks

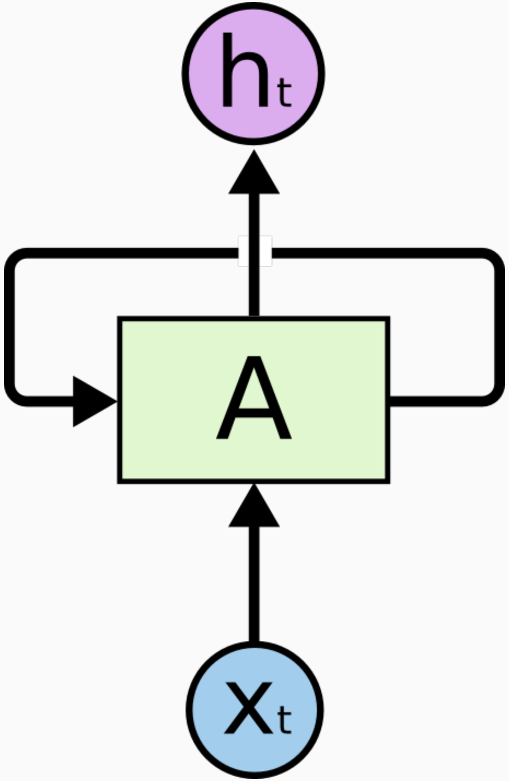
Learn more about Convolutional Neural Networks

Recurrent Neural Networks

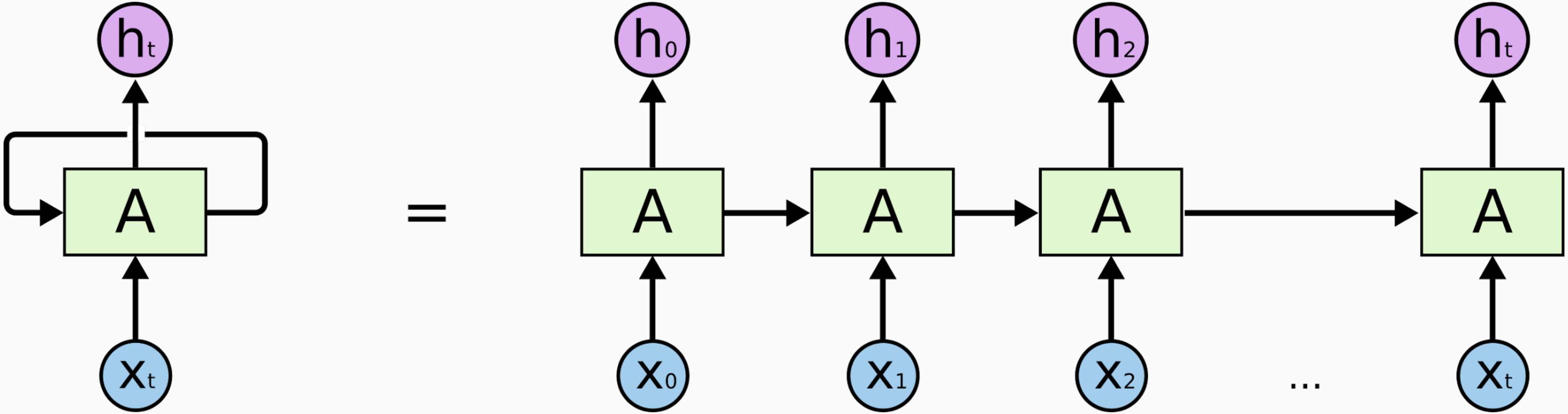
The Idea: Loops

Recurrent Neural Networks have **loops** - they are fed their own output as an input in the next time step.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/



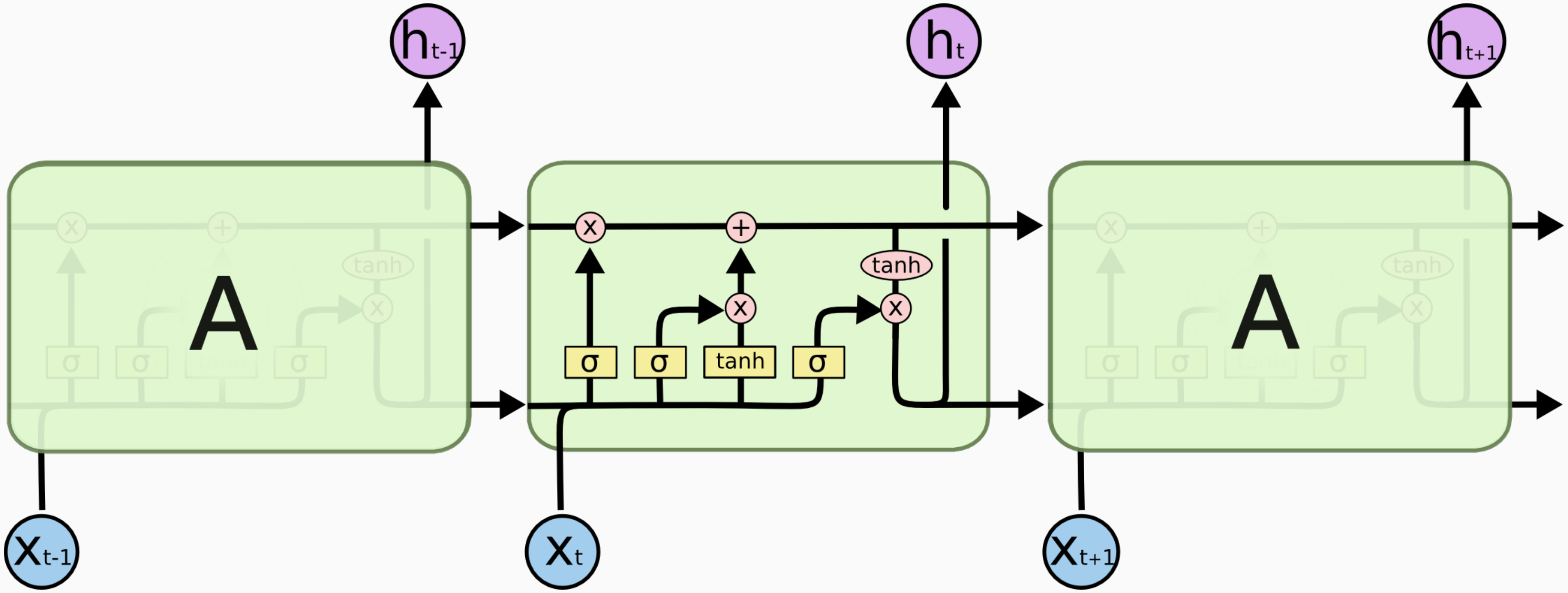
The Idea: Unrolling a RNN in time



A RNN can be **unrolled** in time to obtain a feedforward neural network.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The Idea: What happens in a RNN Cell stays in a RNN cell.



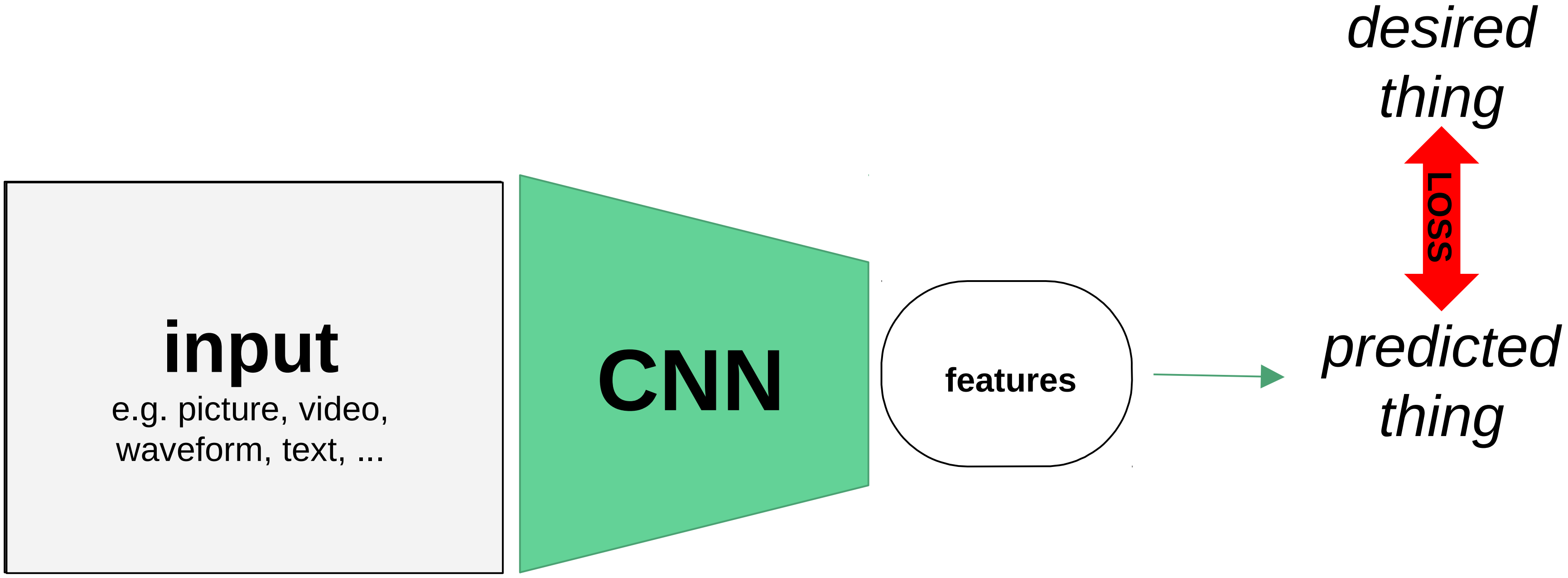
Schematic diagram of a Long Short-Term Memory Network.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Applications

II.) Applications

Specify some task...



Specify some task...

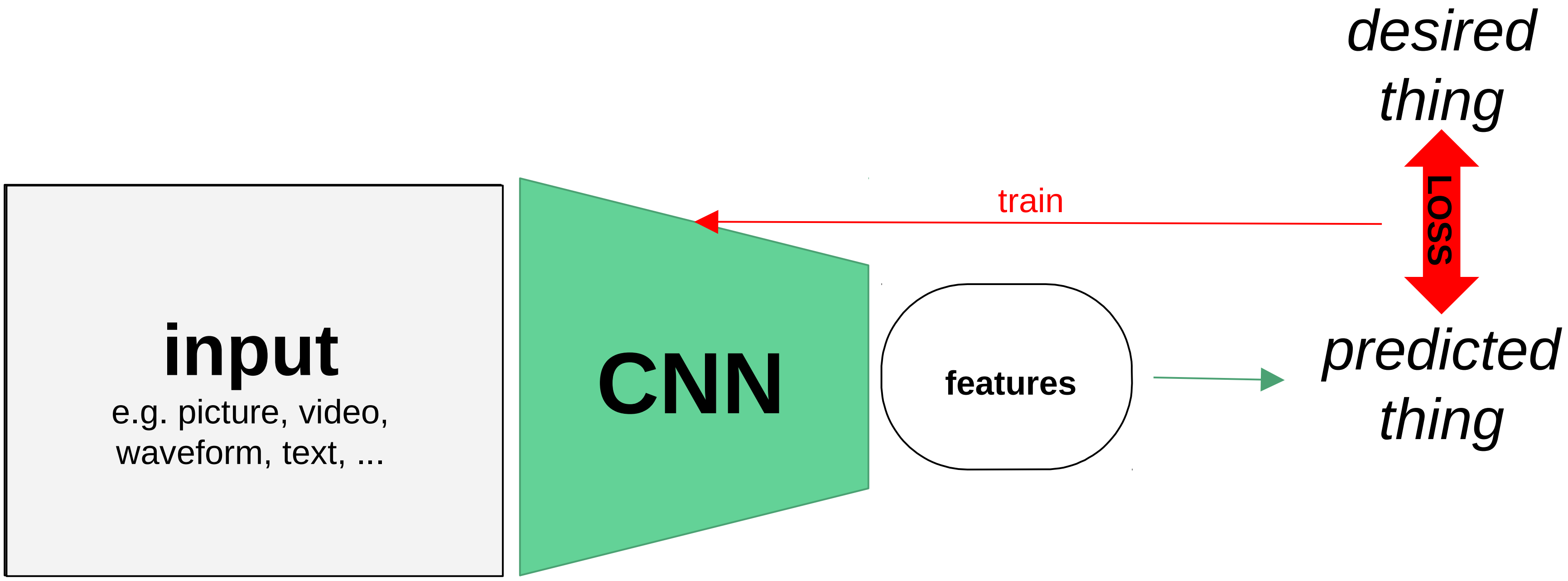
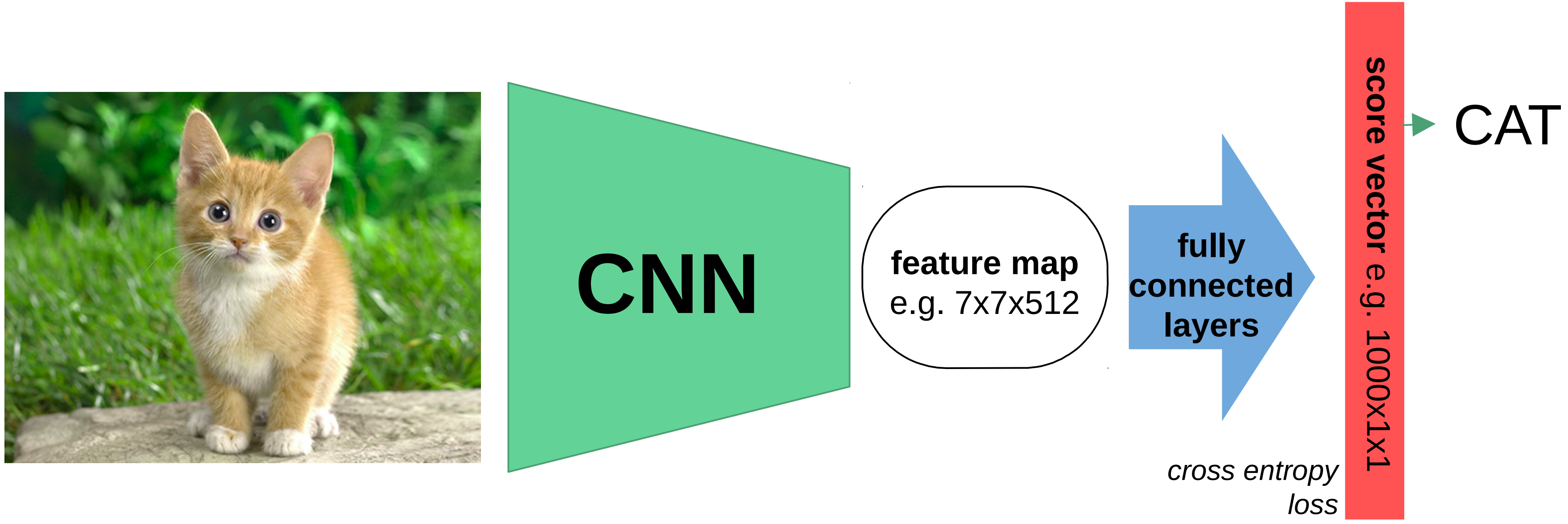
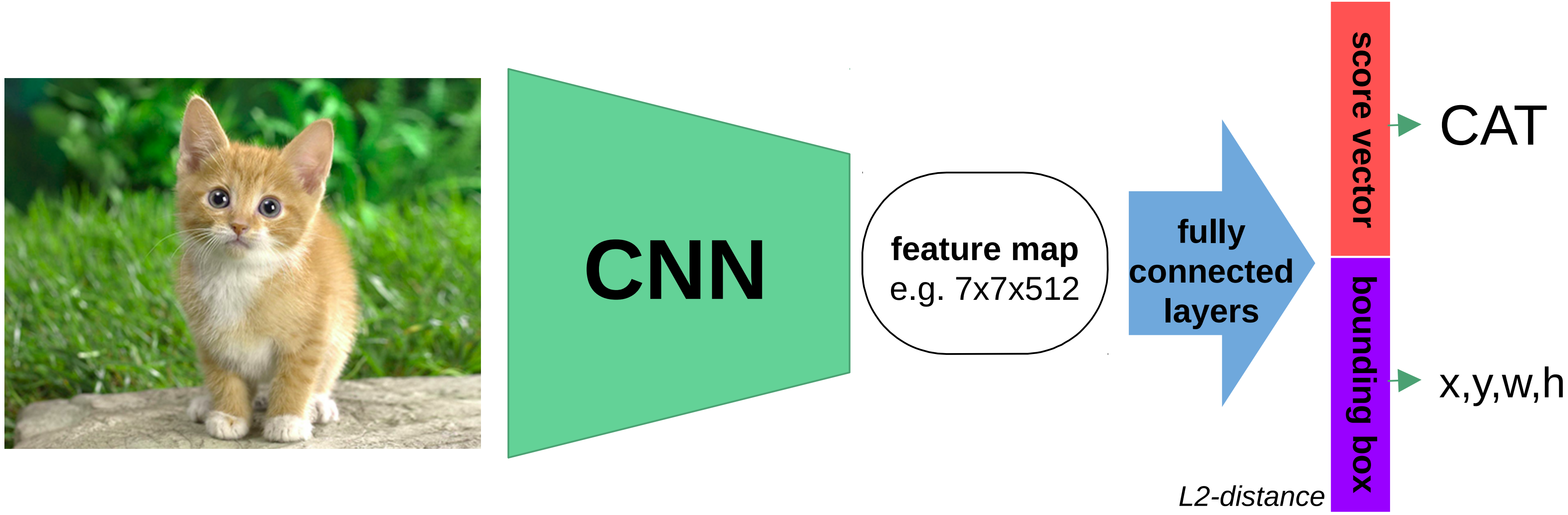


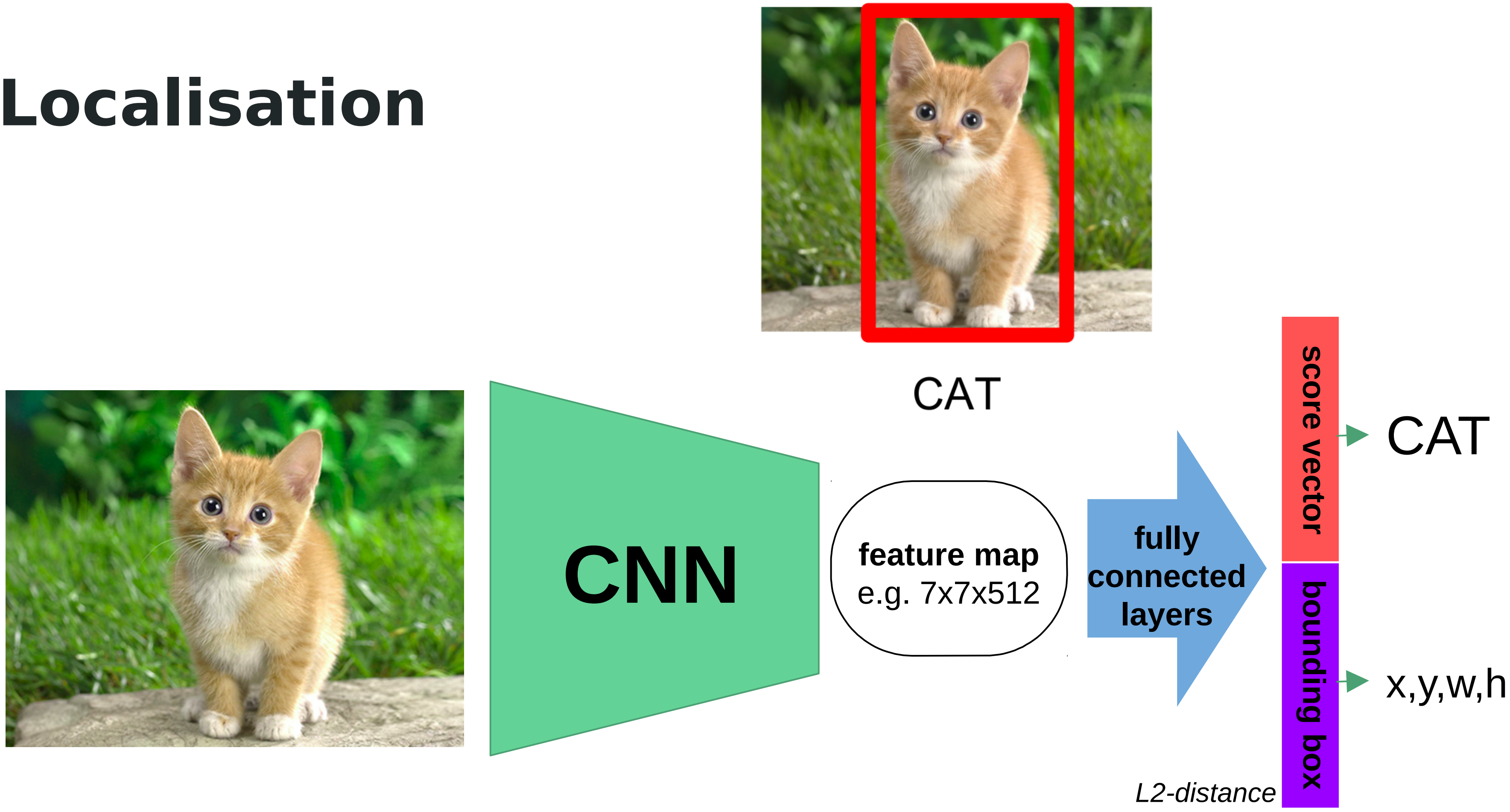
Image Classification



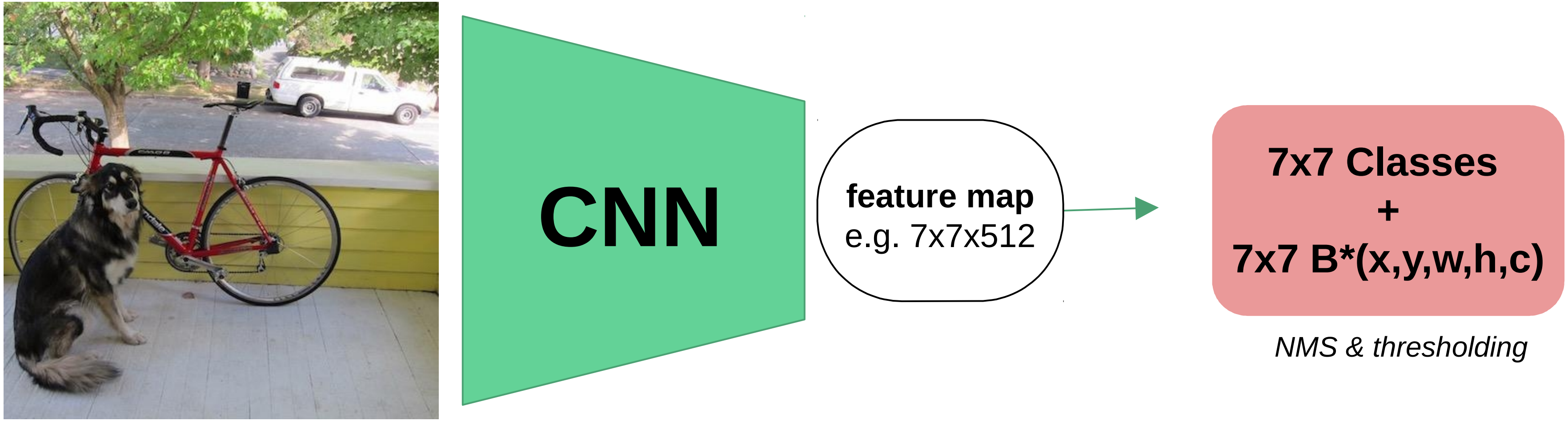
Localisation



Localisation

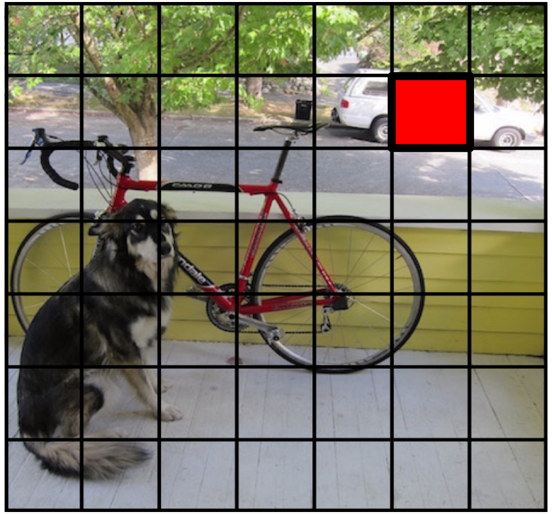


Detection

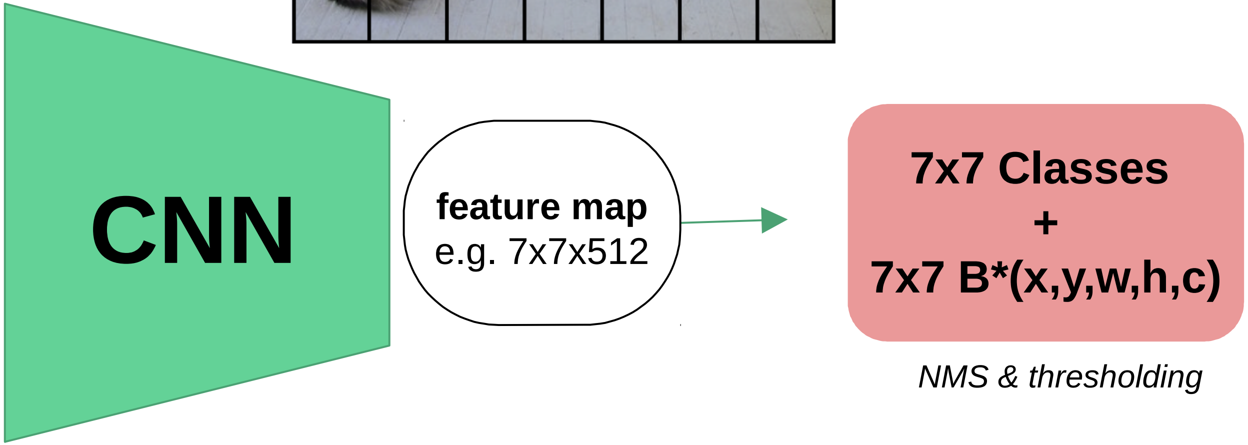


e.g. YOLO - Redmon et al, 2016

Detection



e.g. YOLO - Redmon et al, 2016



NMS & thresholding

Detection

dog
bicycle
car
dining table

e.g. YOLO - Redmon et al, 2016

CNN

feature map
e.g. 7x7x512

7x7 Classes
+
7x7 B*(x,y,w,h,c)

NMS & thresholding

Detection

e.g. YOLO - Redmon et al, 2016

CNN

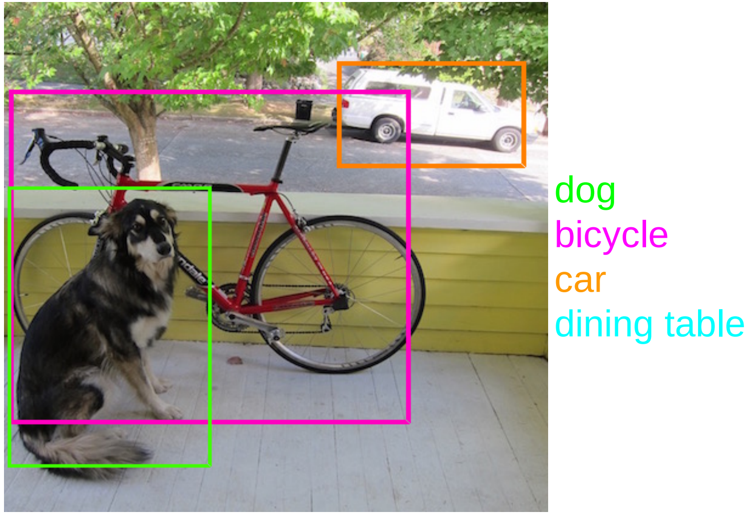
feature map
e.g. 7x7x512

7x7 Classes
+
7x7 B*(x,y,w,h,c)

NMS & thresholding

dog
bicycle
car
dining table

Detection



e.g. YOLO - Redmon et al, 2016

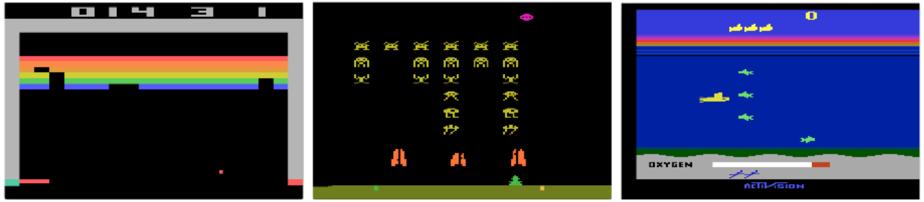
NMS & thresholding

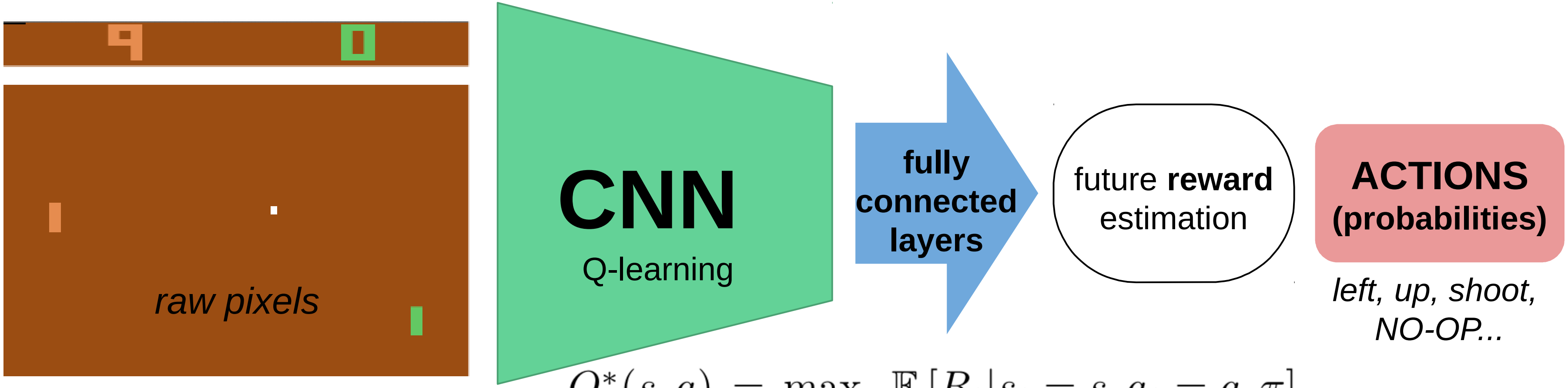
Detection

Redmon
et al, 2017

pjreddie.com/
darknet/yolo/

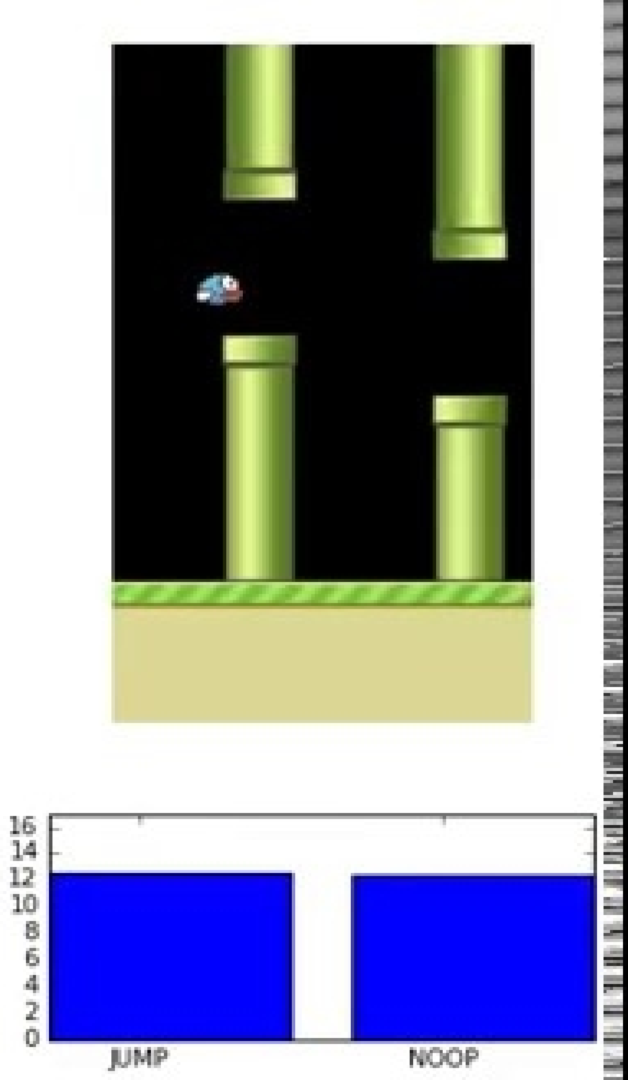
Reinforcement Learning





$$Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

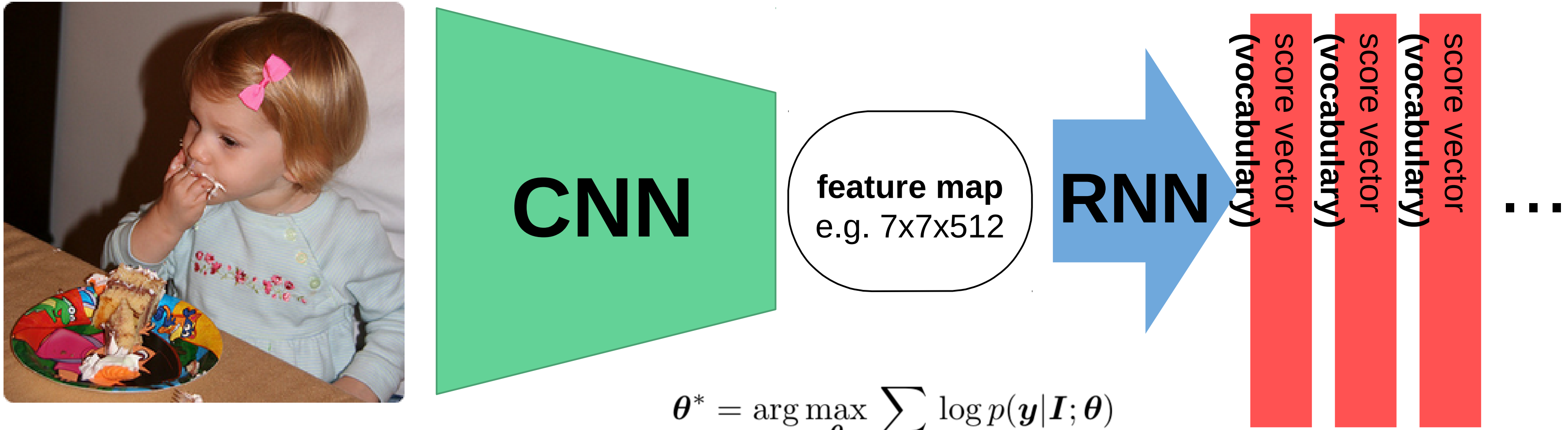
Mnih et al, 2013



gym.openai.com

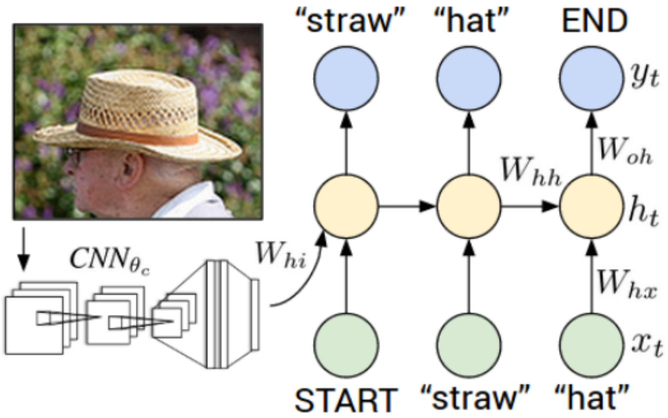
credit:
Lukas Palm

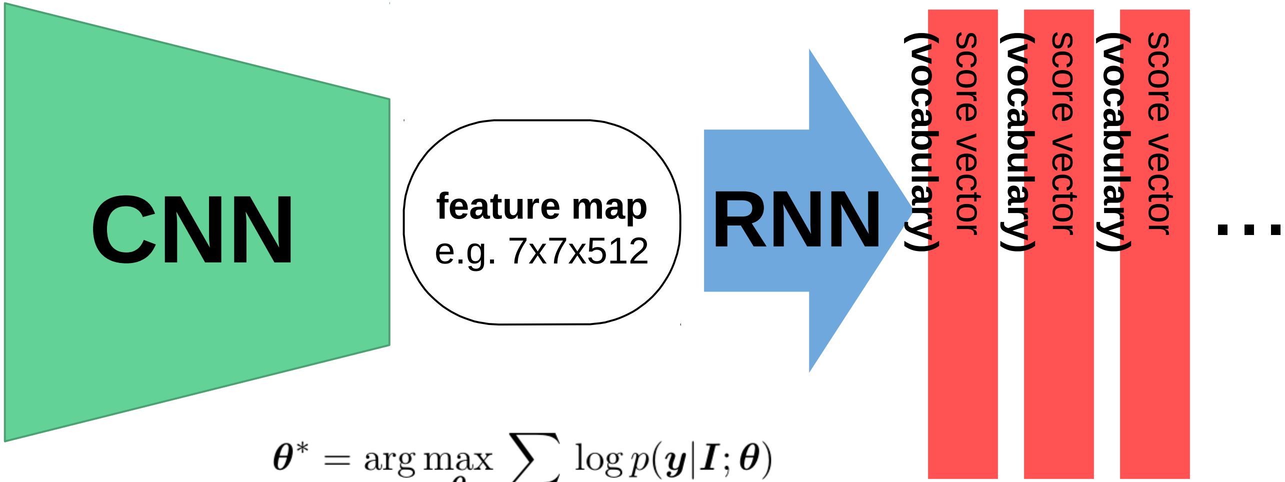
Image Captioning



$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{I},\boldsymbol{y})} \log p(\boldsymbol{y}|\boldsymbol{I};\boldsymbol{\theta})$$

Image Captioning





$$\theta^* = \arg\max_{\theta} \sum_{(I,y)} \log p(y|I;\theta)$$

Image Captioning

Karpathy et al, 2015

"little girl is eating piece of cake."

"baseball player is throwing ball in game."

"woman is holding bunch of bananas."

"black cat is sitting on top of suitcase."

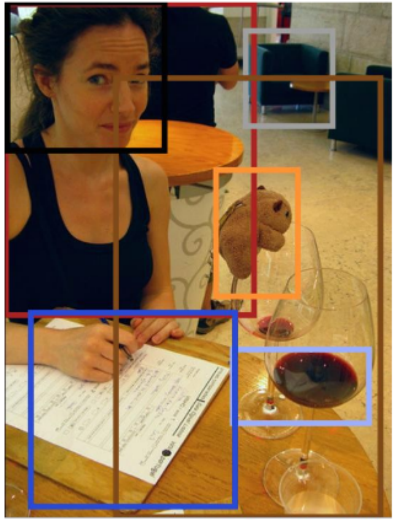
"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."

"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."

Dense Image Captioning and Attention

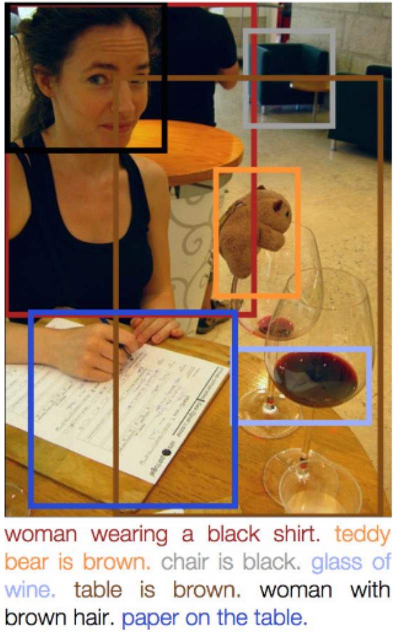


woman wearing a black shirt. teddy bear is brown. chair is black. glass of wine. table is brown. woman with brown hair. paper on the table.

A man and a woman sitting at a table with a cake.

Johnson et al, 2015

Dense Image Captioning and Attention

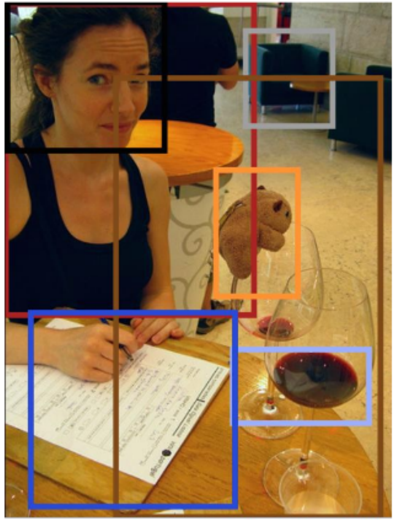


A man and a woman sitting at a table with a cake.

Johnson et al, 2015

Lu et al, 2016

Dense Image Captioning and Attention

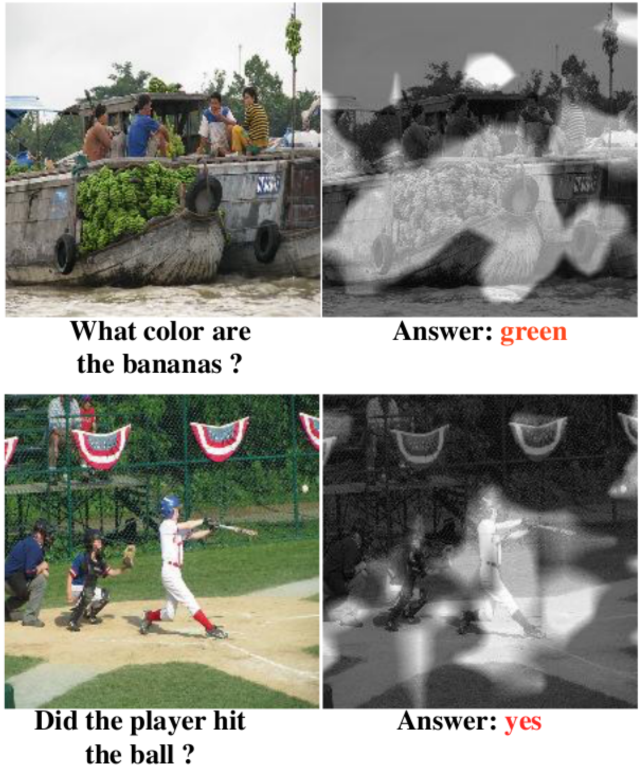


woman wearing a black shirt. teddy bear is brown. chair is black. glass of wine. table is brown. woman with brown hair. paper on the table.

A man and a woman sitting at a table with a cake.

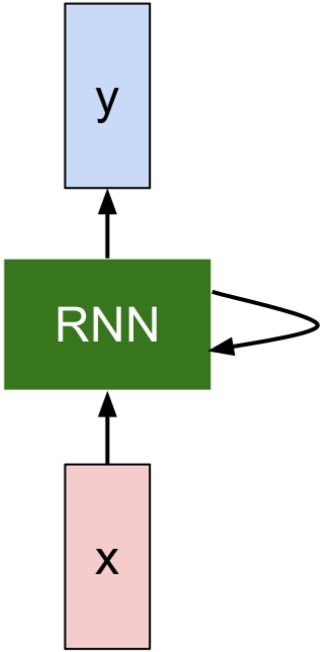
Johnson et al, 2015

Lu et al, 2016



Socher, 2016 `youtube.com/watch?v=oGk1v1jQITw`

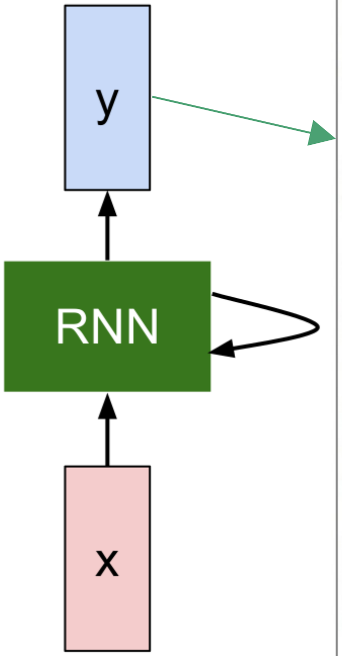
RNNs: Generating Algebraic Geometry



`stacks.math.columbia.edu/`

16MB LaTex file

RNNs: Generating Algebraic Geometry



Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

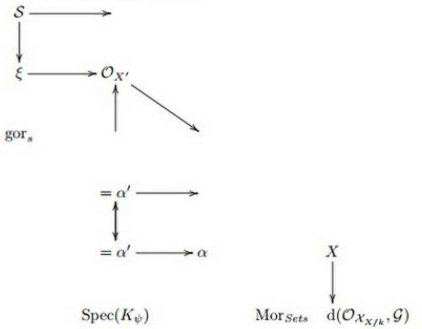
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

1. $\mathcal{F}$ is an algebraic space over S.
2. If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{étale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

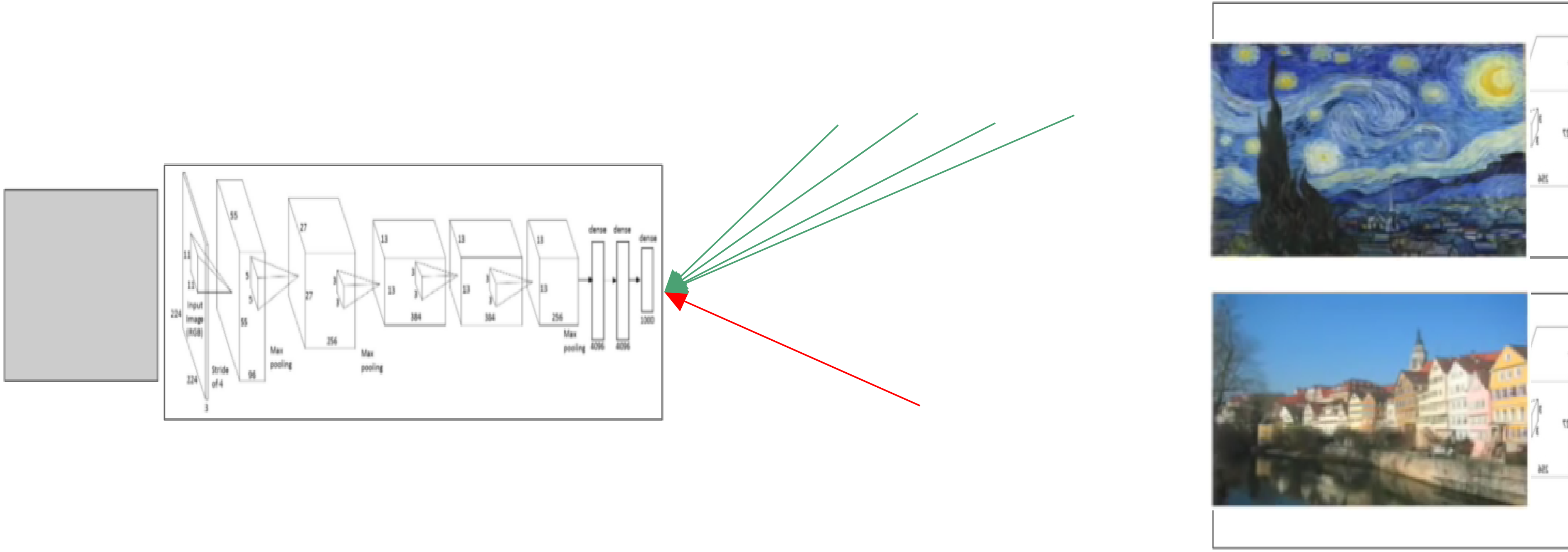
is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

stacks.math.columbia.edu/
16MB LaTex file

karpathy.github.io/2015/05/21/rnn-effectiveness/

Neural Style

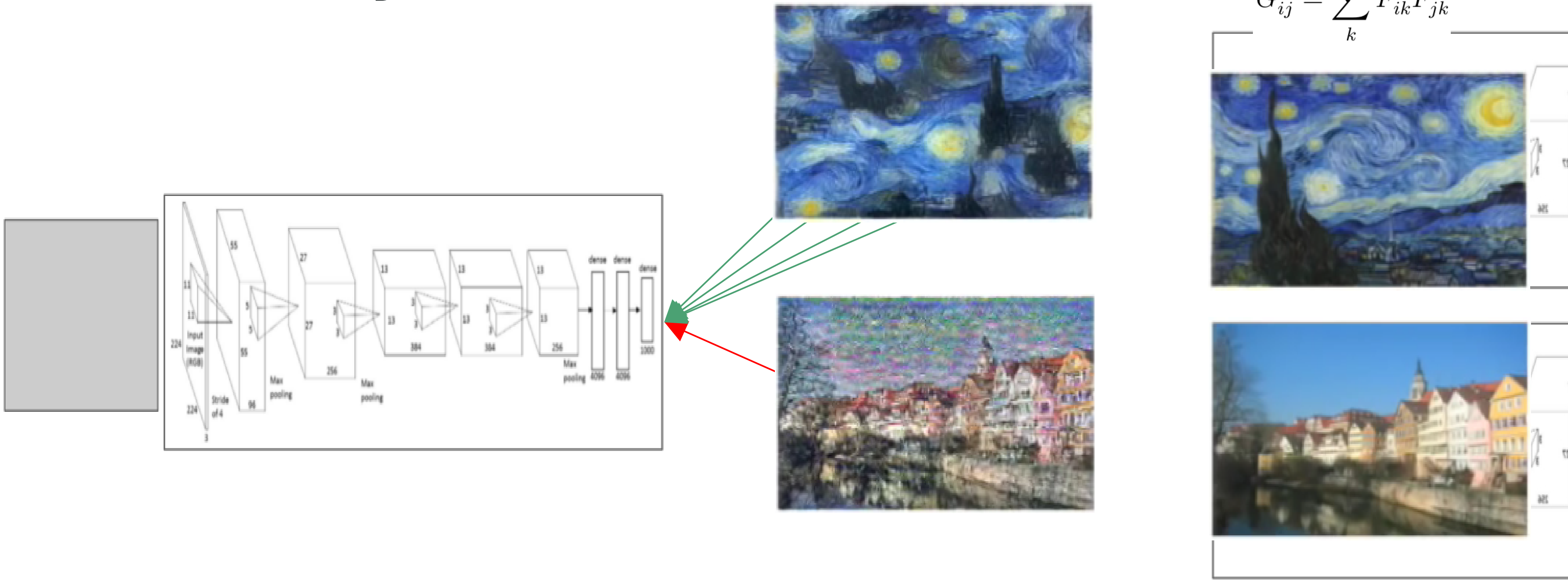


Gatys et al, 2015
deepart.io

Neural Style

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$



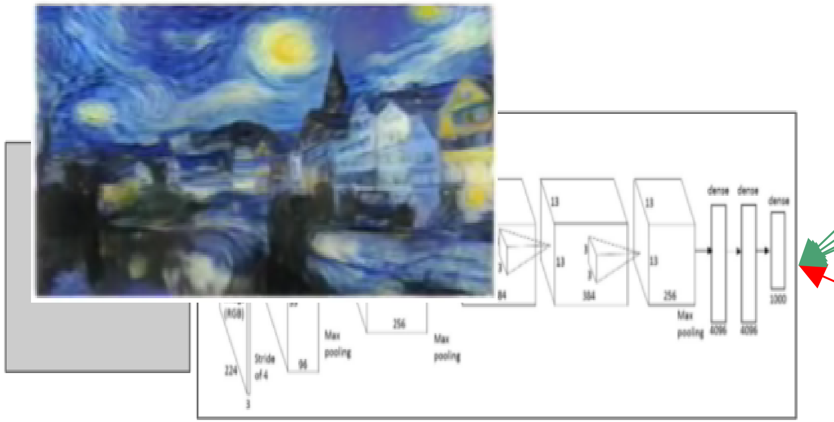
$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

Gatys et al, 2015
deepart.io

Neural Style

$$\mathcal{L}_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

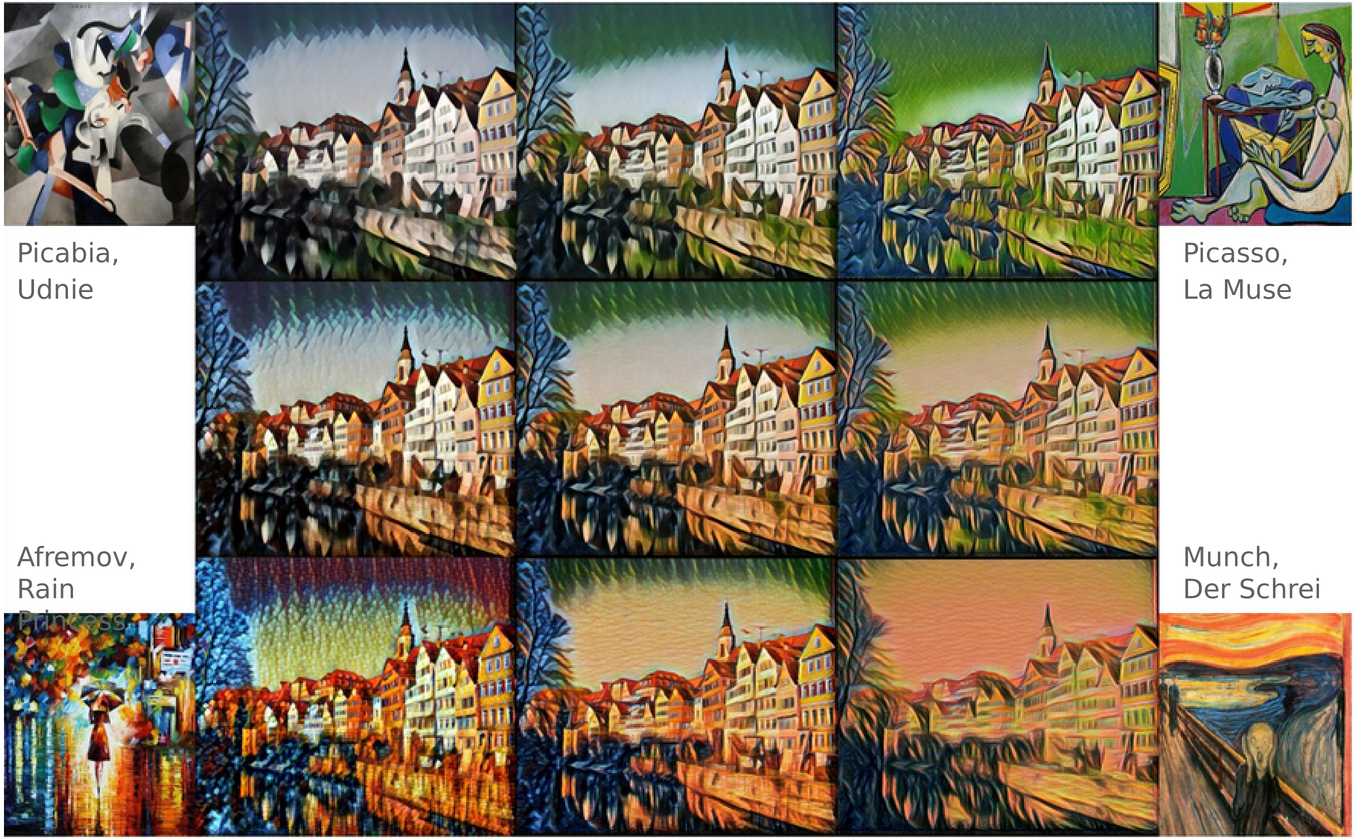


$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x}) = \alpha \mathcal{L}_{content}(\vec{p},\vec{x}) + \beta \mathcal{L}_{style}(\vec{a},\vec{x})$$

$$\mathcal{L}_{content}(\vec{p},\vec{x},l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2$$

Gatys et al, 2015
`deepart.io`

Neural Style - Interpolation



Picabia, Udnie

Picasso, La Muse

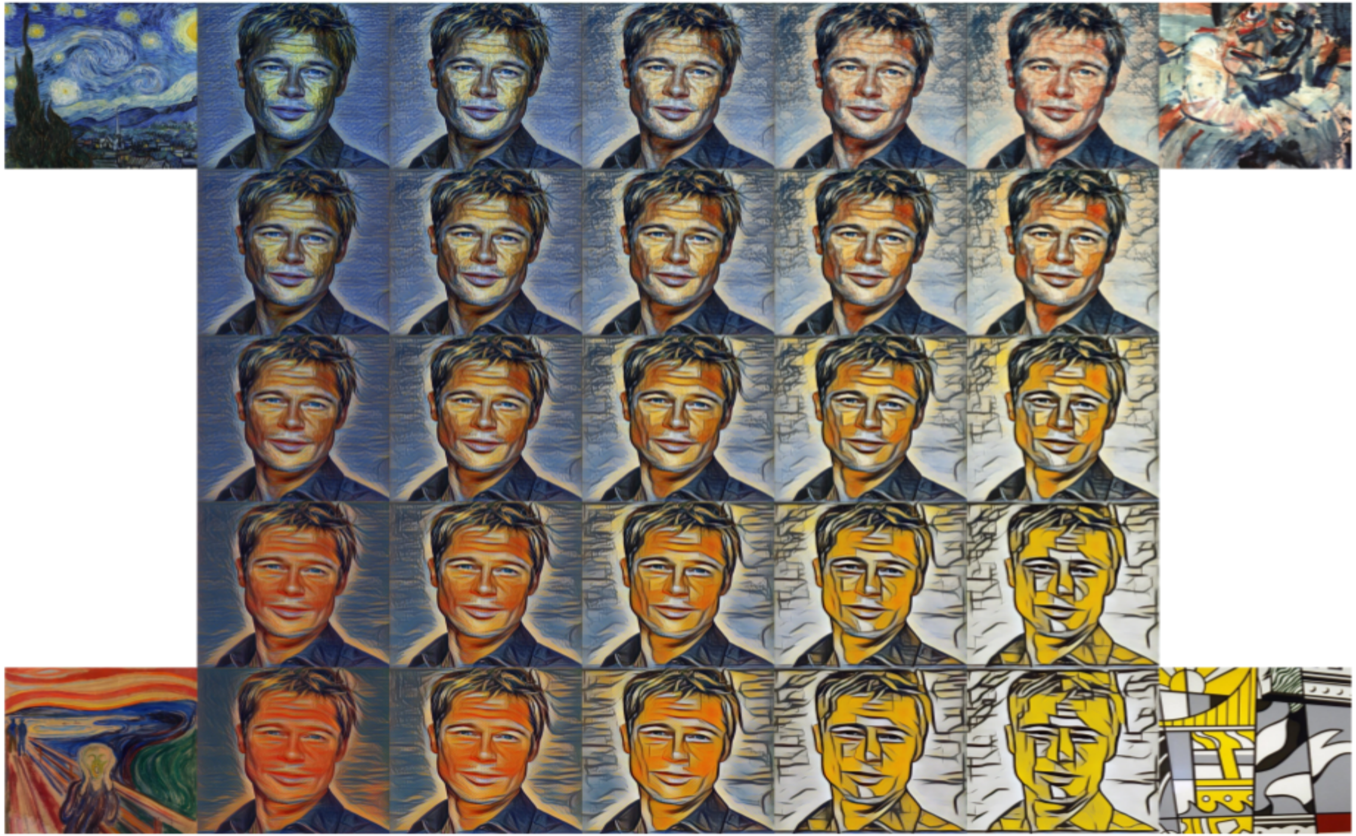
Afremov, Rain Princess

Munch, Der Schrei

github.com/Heumi/Fast_Multi_Style_Transfer-tensorflow

Dumoulin et al, 2016

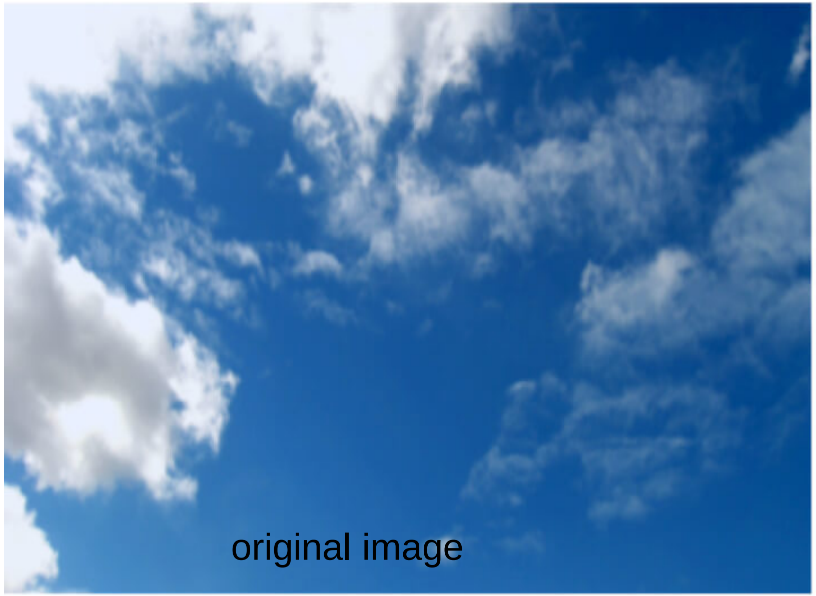
Neural Style - Interpolation

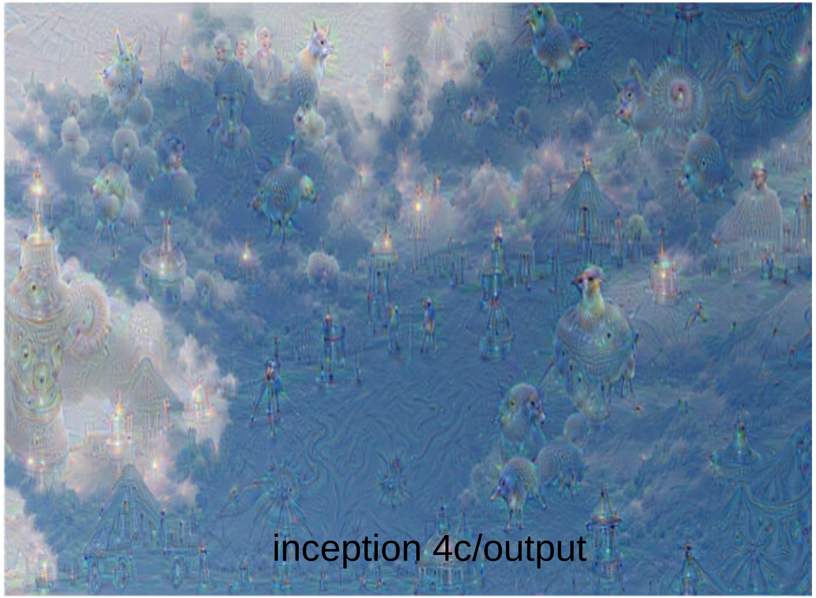


github.com/Heumi/Fast_Multi_Style_Transfer-tensorflow

Dumoulin et al, 2016

Deep Dream





github.com/google/deepdream

```
net.forward(end=end)
objective(dst)
net.backward(start=end)
```

```
def objective_L2(dst):
    dst.diff[:] = dst.data
```

Deep Dream

"Admiral Dog!" "The Pig-Snail" "The Camel-Bird" "The Dog-Fish"

github.com/google/deepdream

```
net.forward(end=end)
objective(dst)
net.backward(start=end)
```

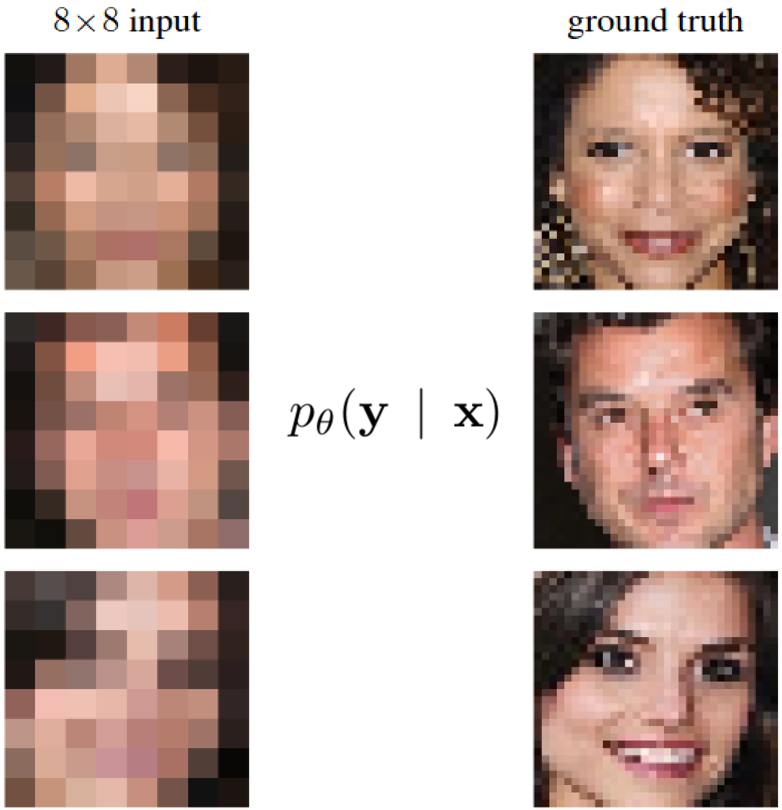
```
def objective_L2(dst):
    dst.diff[:] = dst.data
```

Deep Dream

youtube.com/watch?v=DgPaCWJL7XI

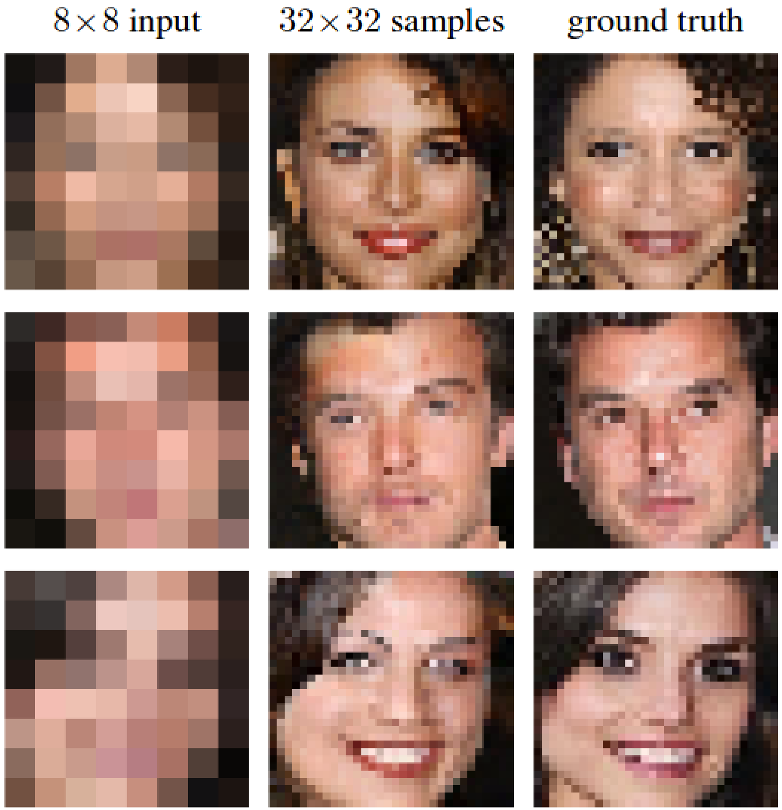
github.com/samim23/DeepDreamAnim

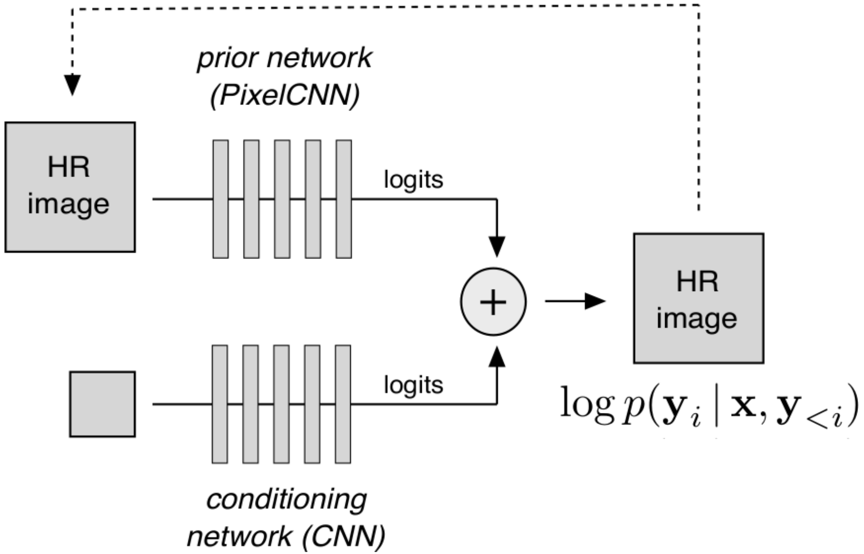
Super Resolution



Dahl et al, 2017

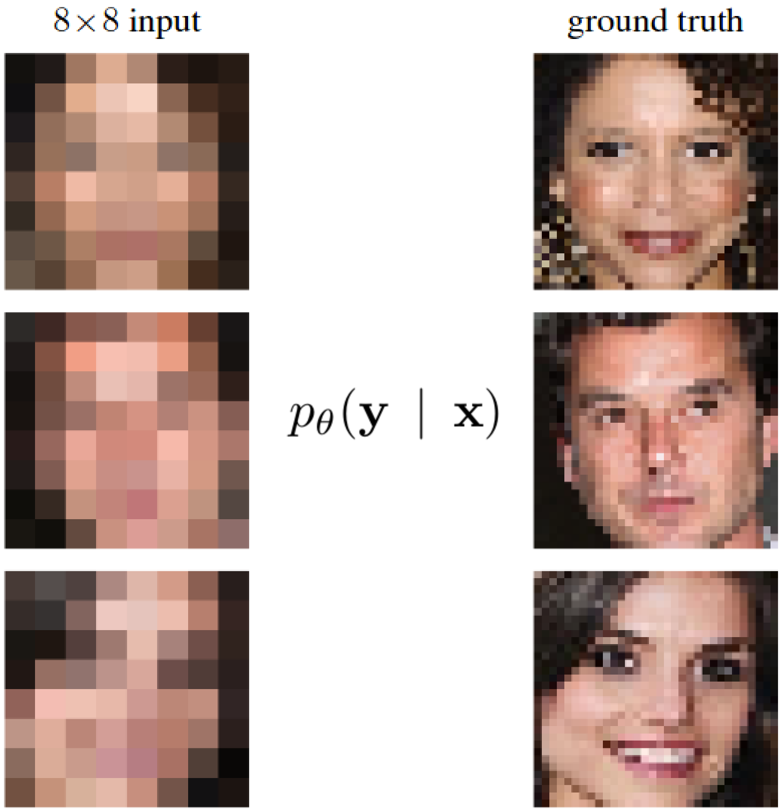
Super Resolution



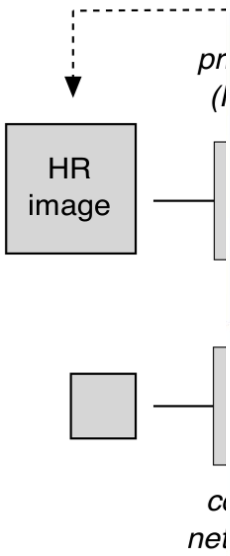


Dahl et al, 2017

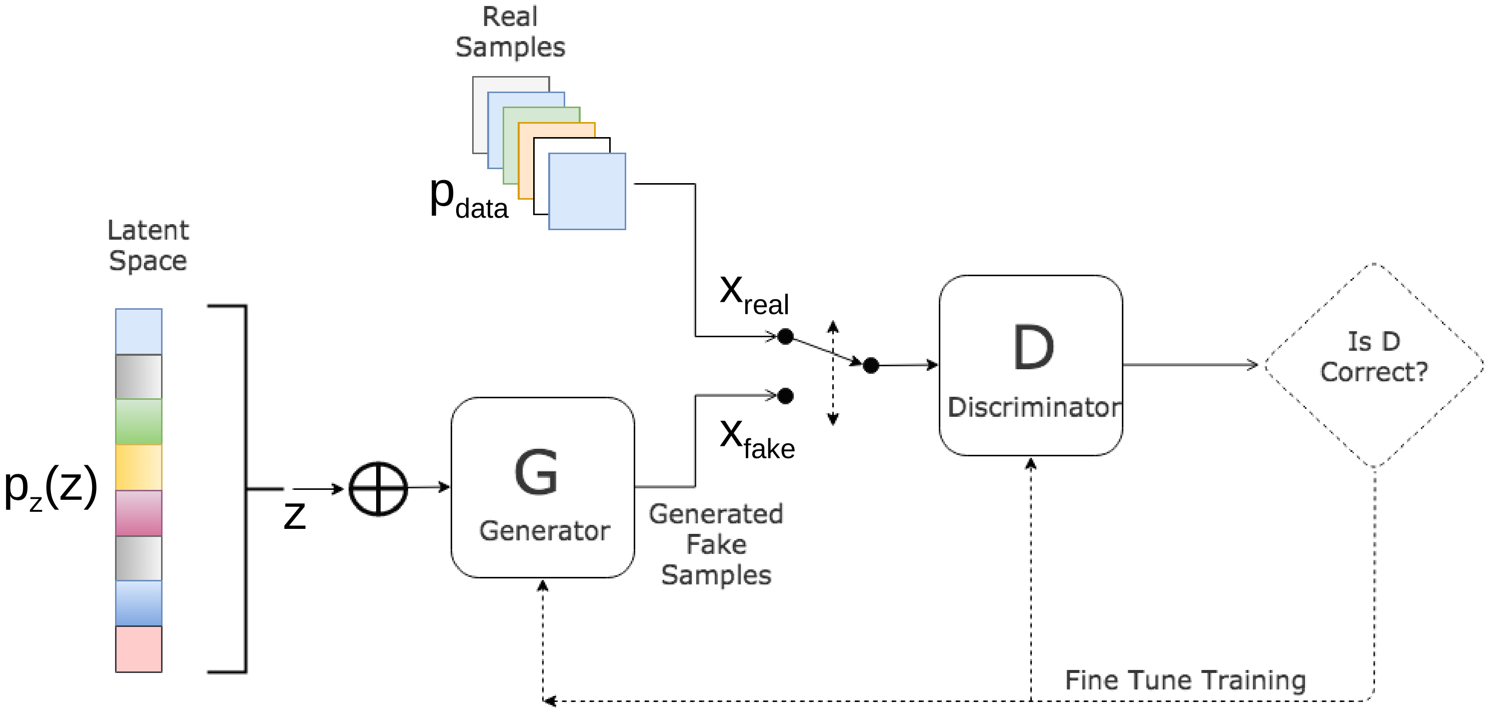
Super Resolution



Dahl et al, 2017



Generative Adversarial Networks



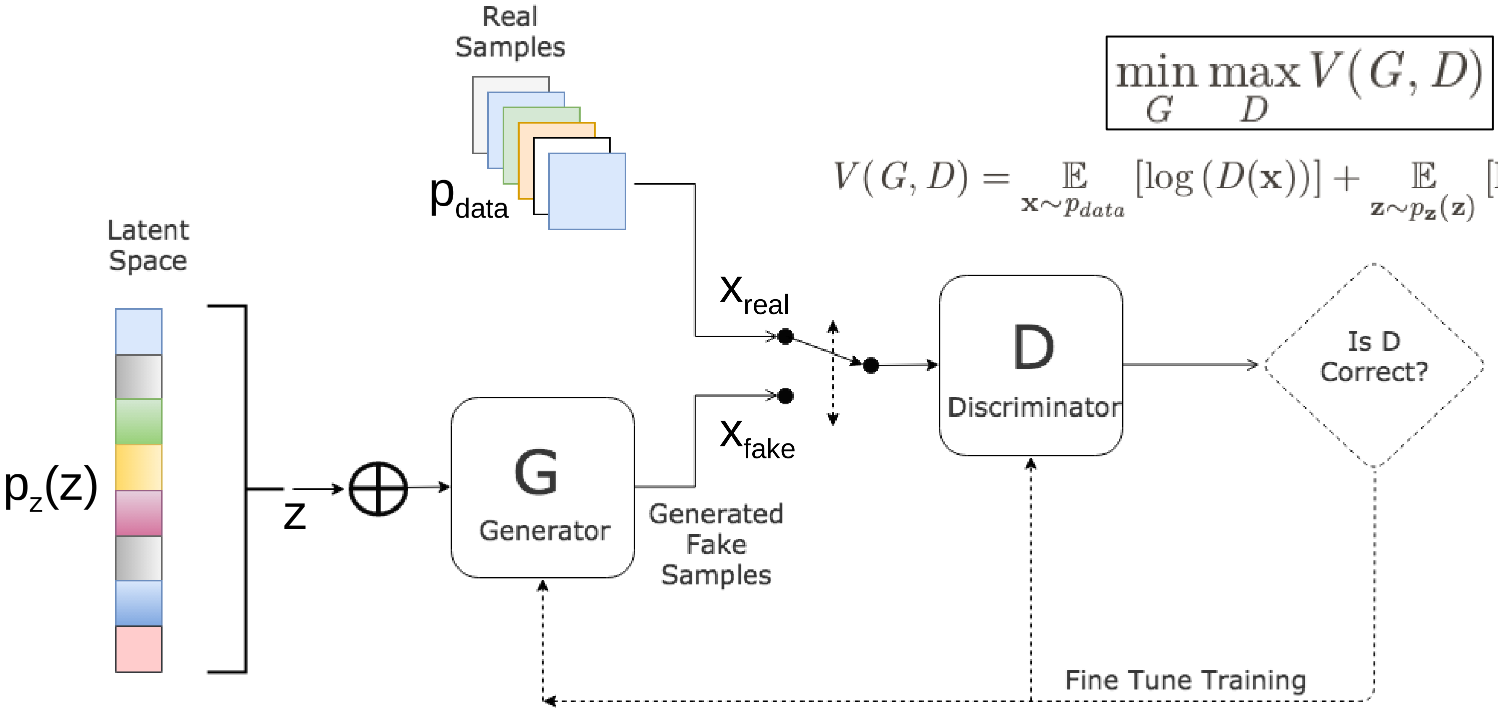
Goodfellow
et al, 2014

kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

Generative Adversarial Networks

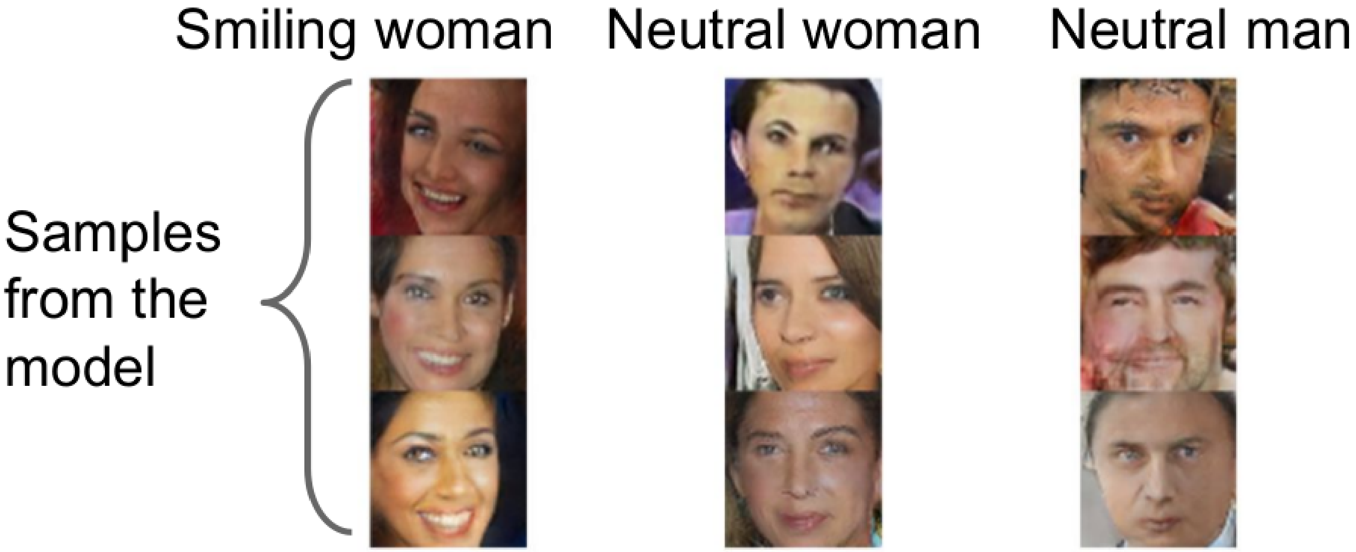
$$\min_G \max_D V(G, D)$$

$$V(G, D) = \mathop{\mathbb{E}}_{\mathbf{x} \sim p_{data}} [\log (D(\mathbf{x}))] + \mathop{\mathbb{E}}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log (1 - D(G(\mathbf{z})))]$$

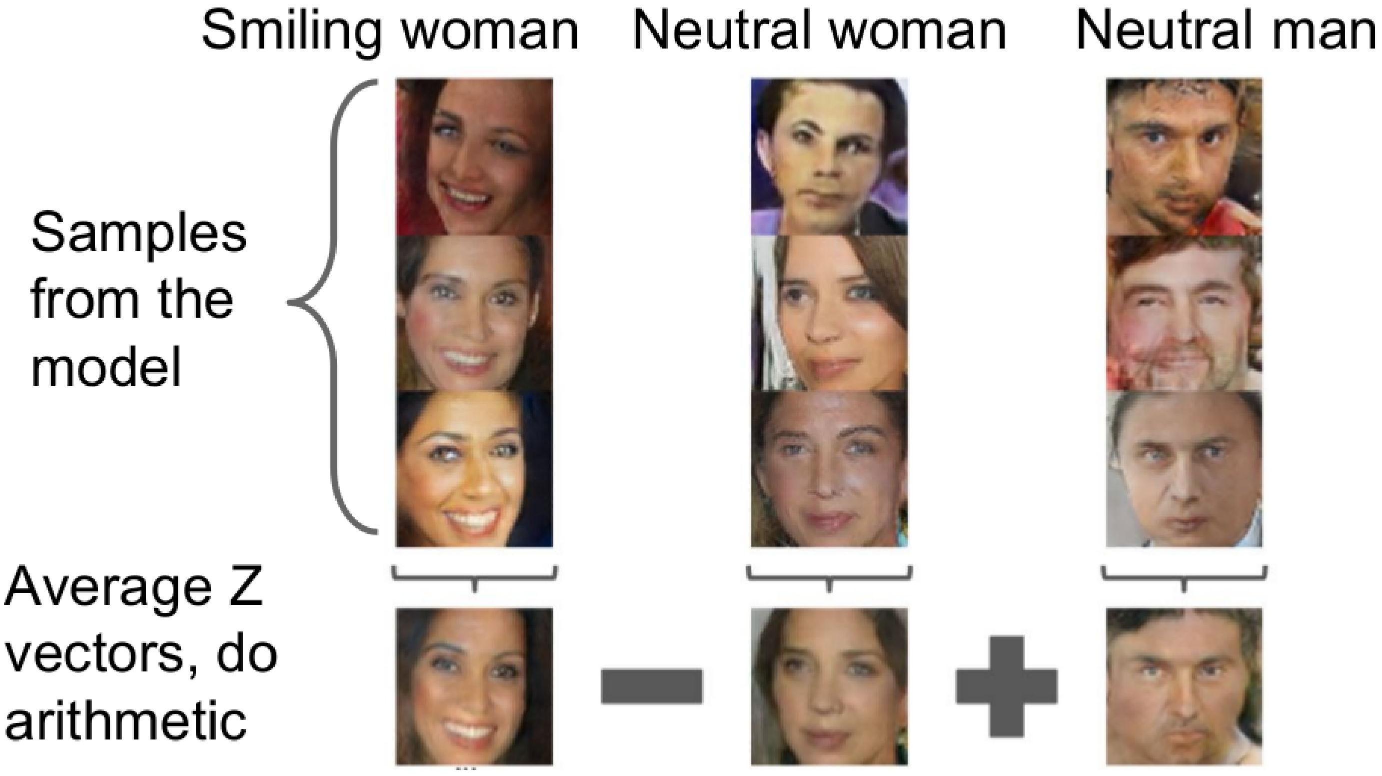


Goodfellow et al, 2014

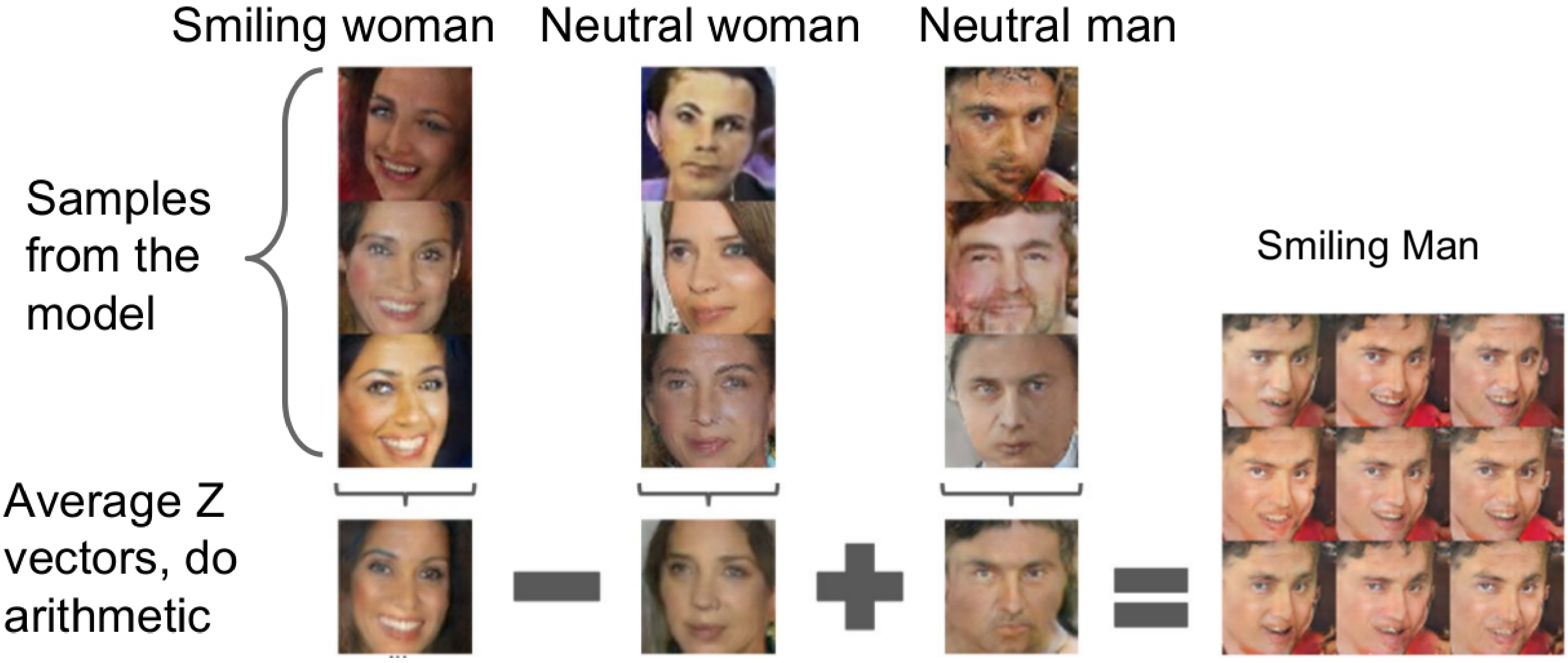
GAN - Results


Smiling woman
Neutral woman
Neutral man
Samples from the model

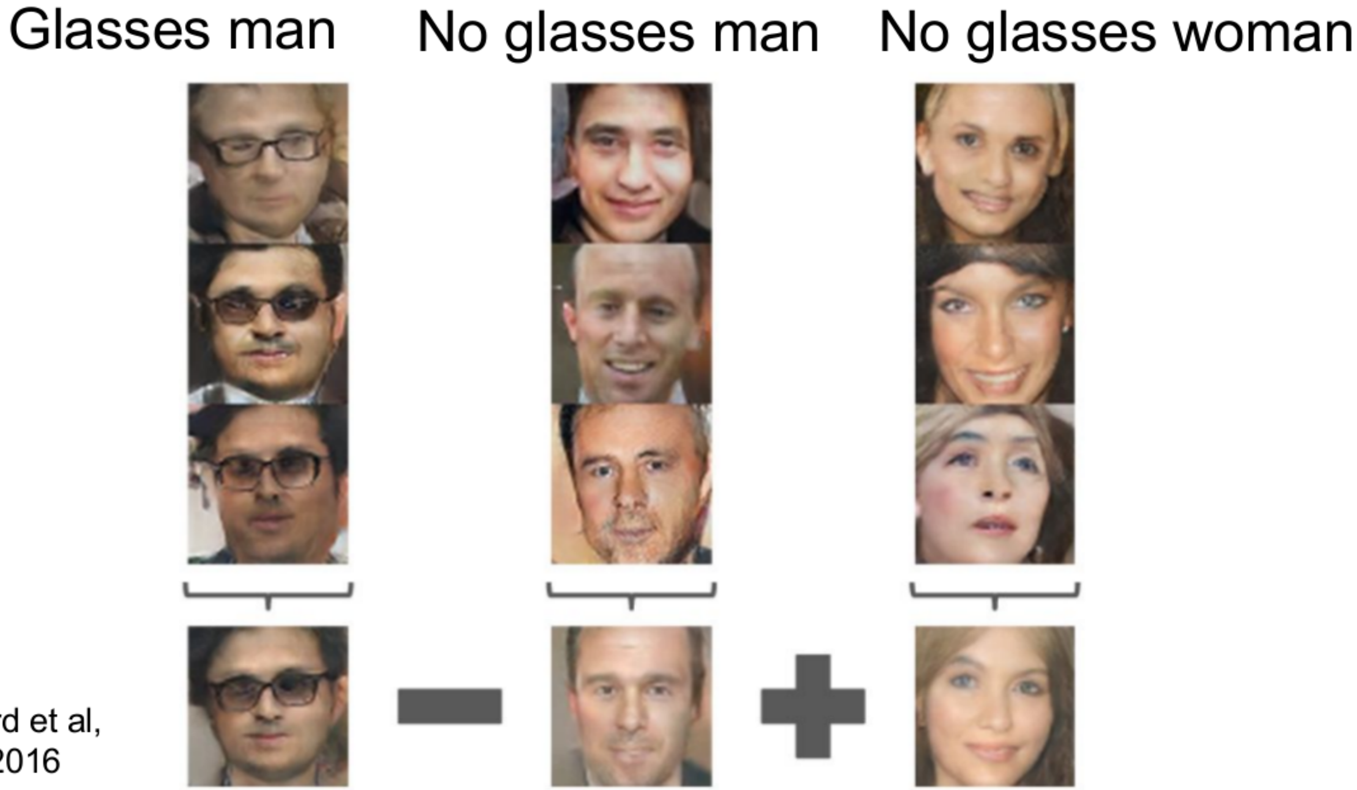
GAN - Vector Arithmetic



GAN - Vector Arithmetic

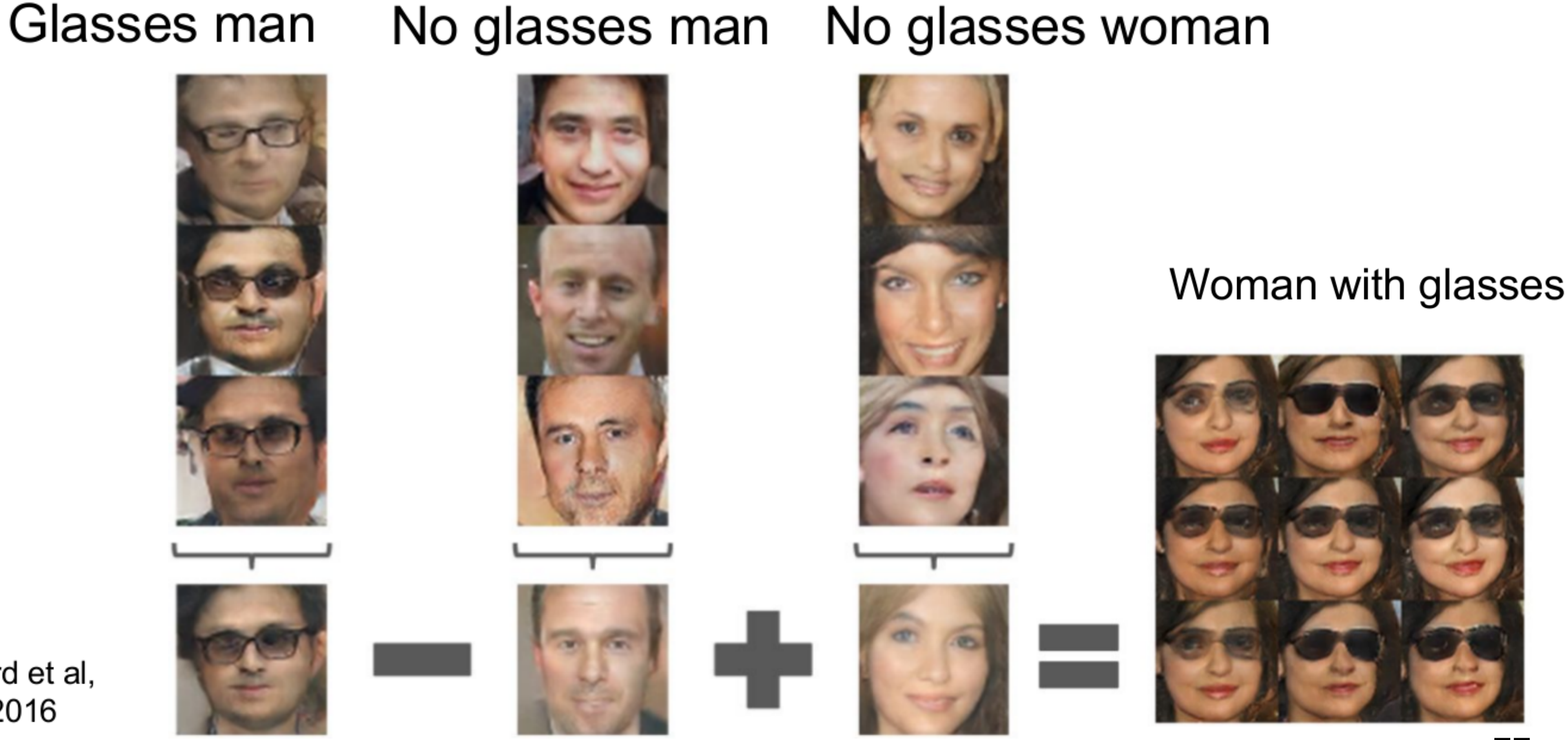


GAN - Vector Arithmetic



Radford et al,
ICLR 2016

GAN - Vector Arithmetic



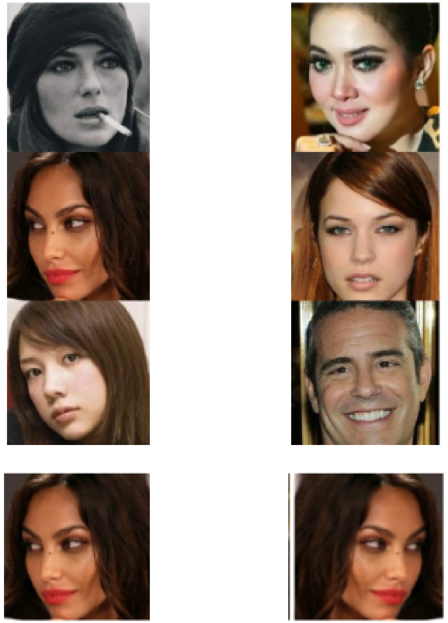
Radford et al,
ICLR 2016

BEGAN: Boundary Equilibrium GAN

train D & G on celebrity faces

find z that matches new images

linear interpolation in latent space



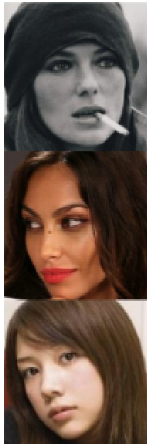
Berthelot et al, 2017

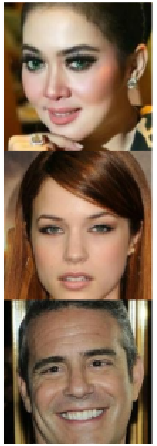
BEGAN: Boundary Equilibrium GAN

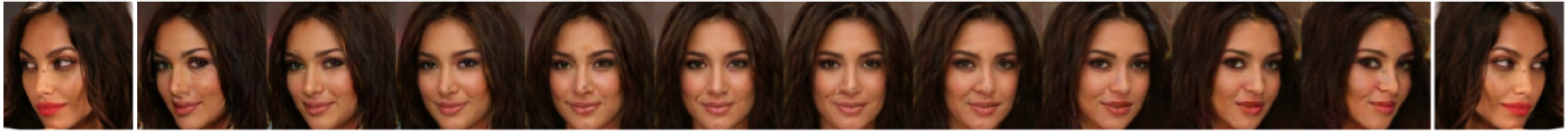
train D & G on celebrity faces

find z that matches new images

linear interpolation in latent space

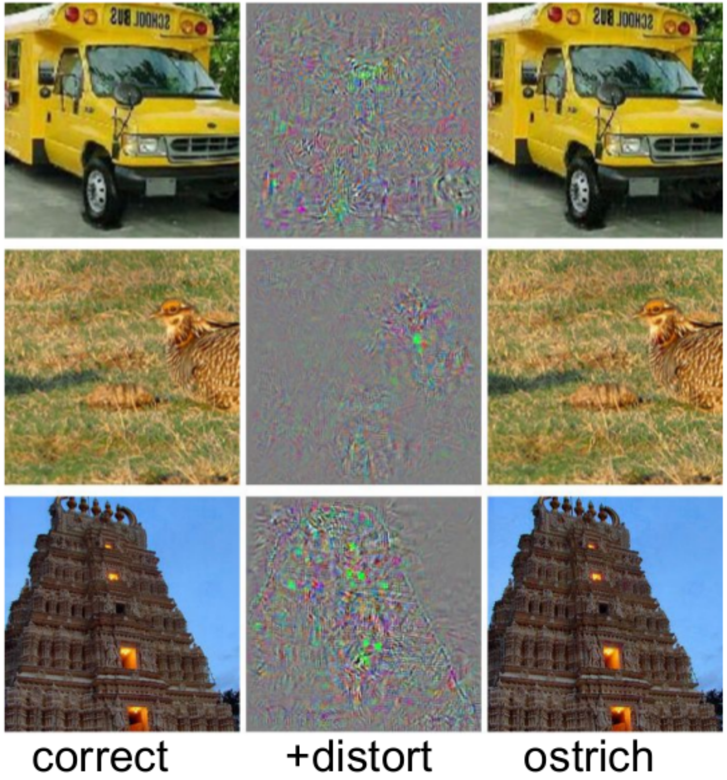






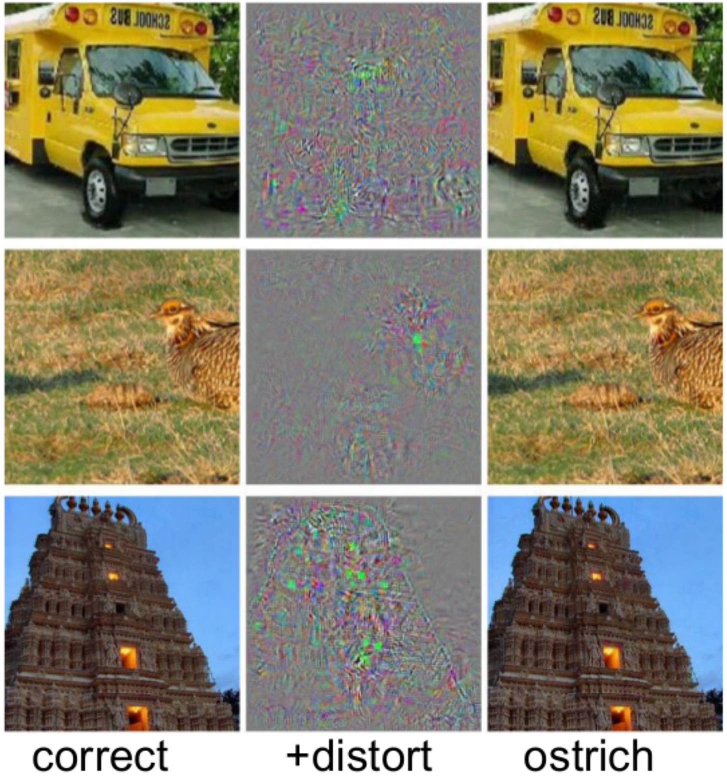
Berthelot et al, 2017

Fooling CNNs

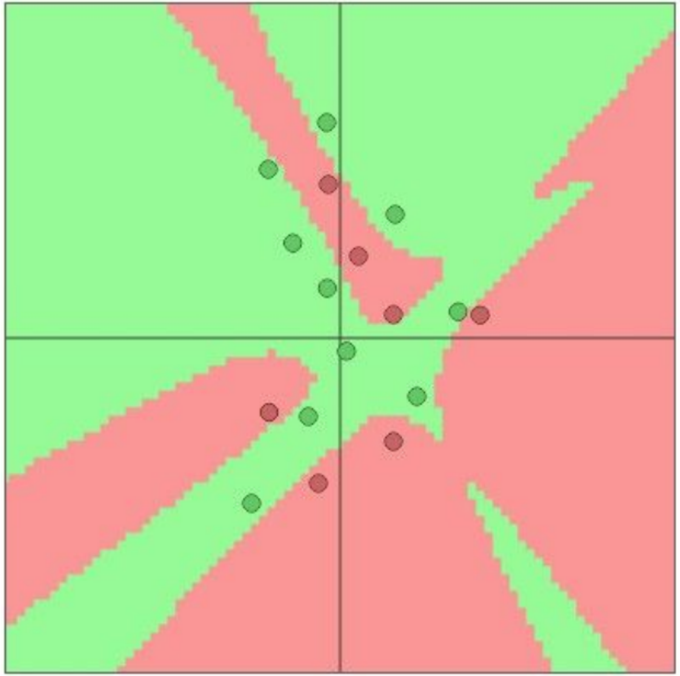


Szegedy et al, 2014

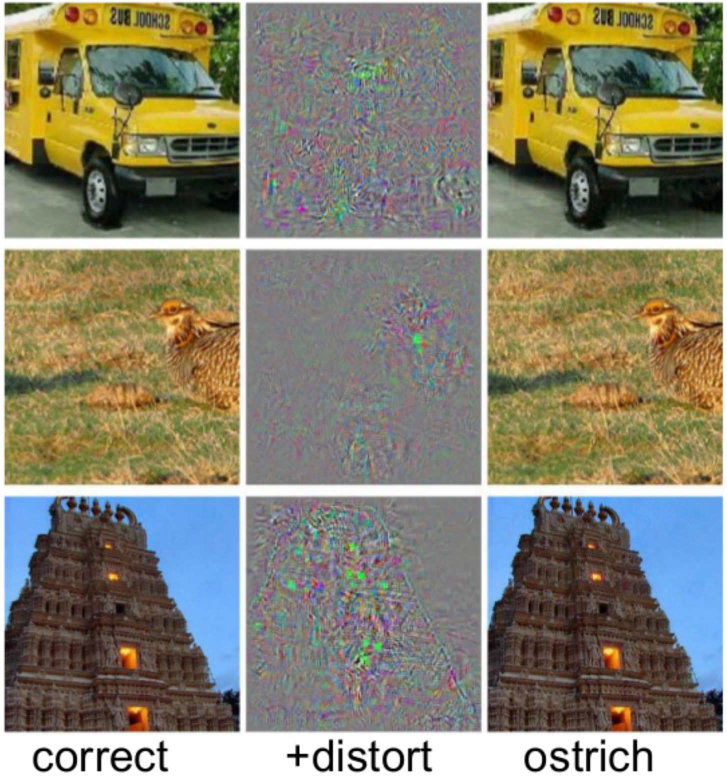
Fooling CNNs



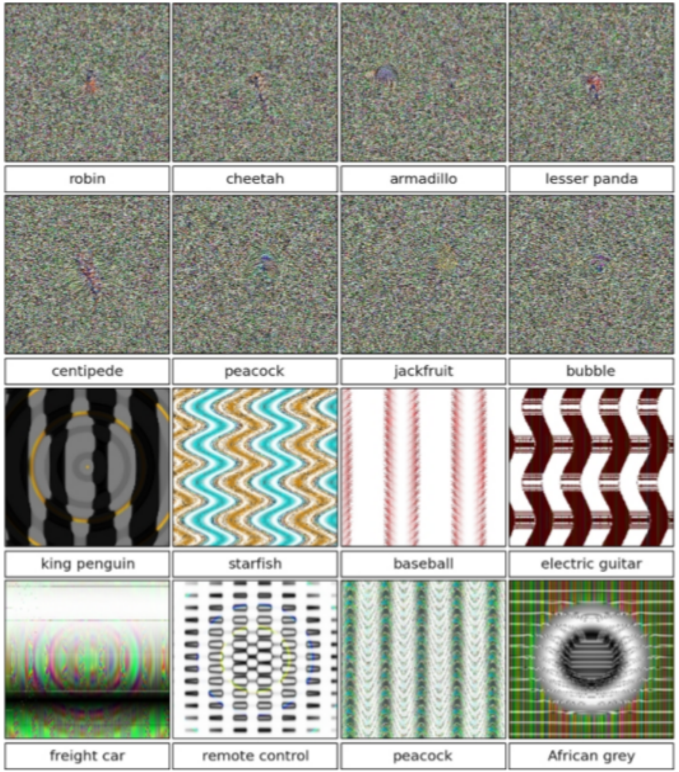
Szegedy et al, 2014



Fooling CNNs

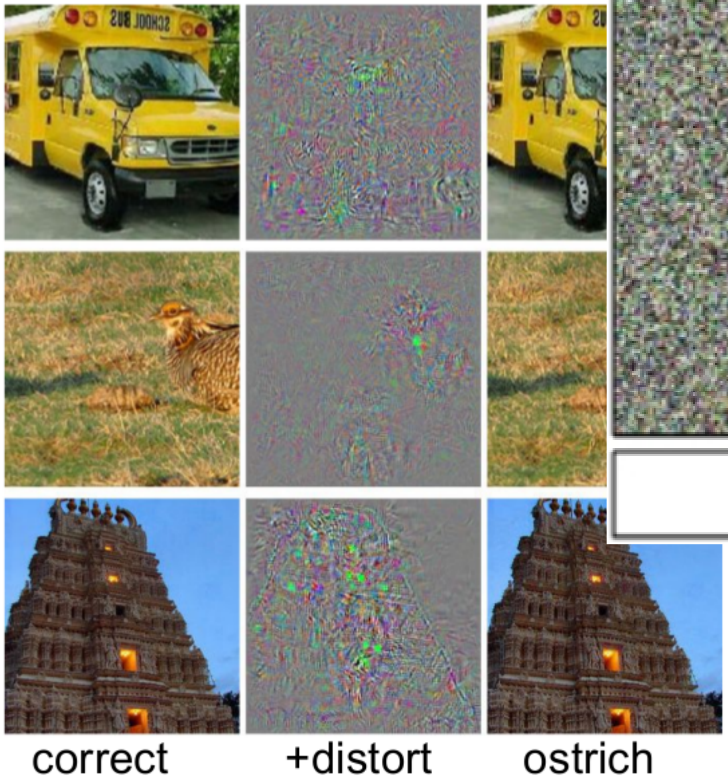


Szegedy et al, 2014

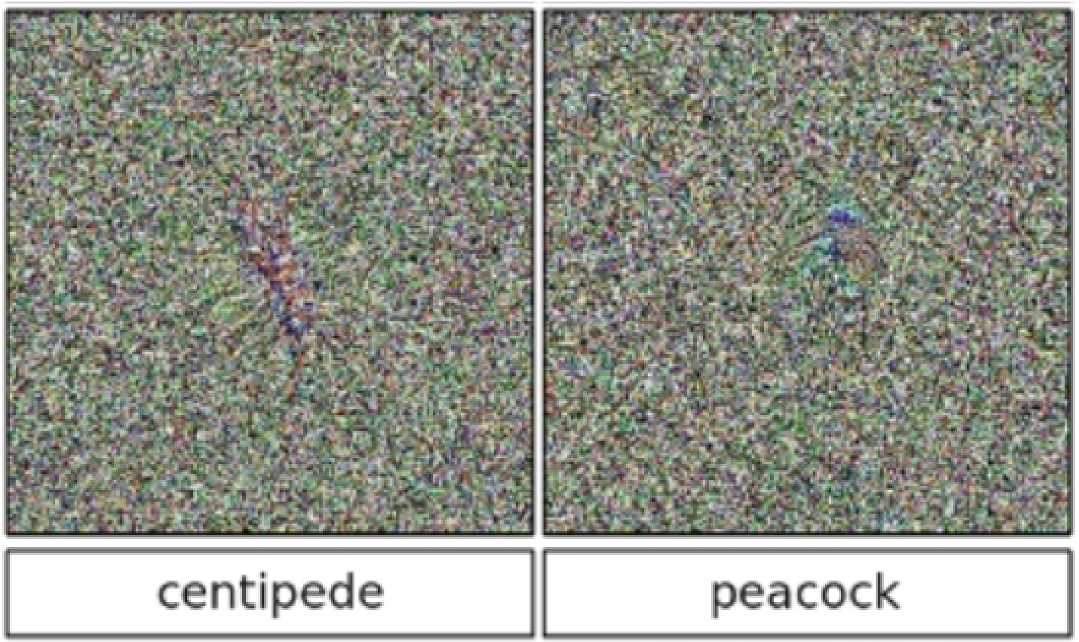


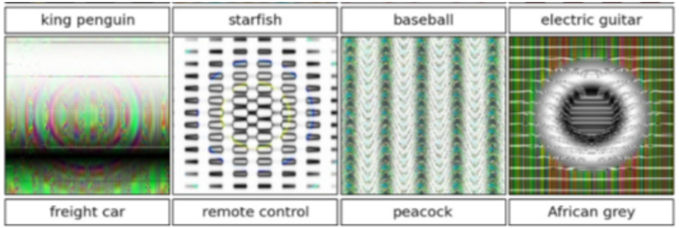
Nguyen et al, 2015

Fooling CNNs



Szegedy et al, 2014





Nguyen et al, 2015

J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” 2016.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013.

A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” 2014.

J. Johnson, A. Karpathy, and L. Fei-Fei, “Densecap: Fully convolutional localization networks for dense captioning,” 2015.

J. Lu, C. Xiong, D. Parikh, and R. Socher, “Knowing when to look: Adaptive attention via a visual sentinel for image captioning,” 2016.

L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” 2015.

V. Dumoulin, J. Shlens, and M. Kudlur, “A learned representation for artistic style,” 2016.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2015.

D. Berthelot, T. Schumm, and L. Metz, “Began: Boundary equilibrium generative adversarial networks,” 2017.

R. Dahl, M. Norouzi, and J. Shlens, “Pixel recursive super resolution,” 2017.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” 2013.

A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” 2014.