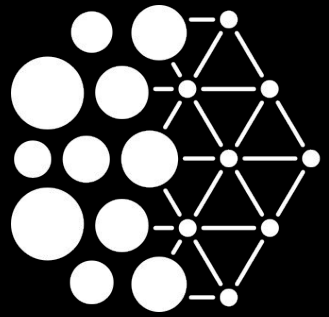
Quebec
Artificial
Intelligence
Institute



Mila

Introduction to Convolutional Neural Networks, Part II

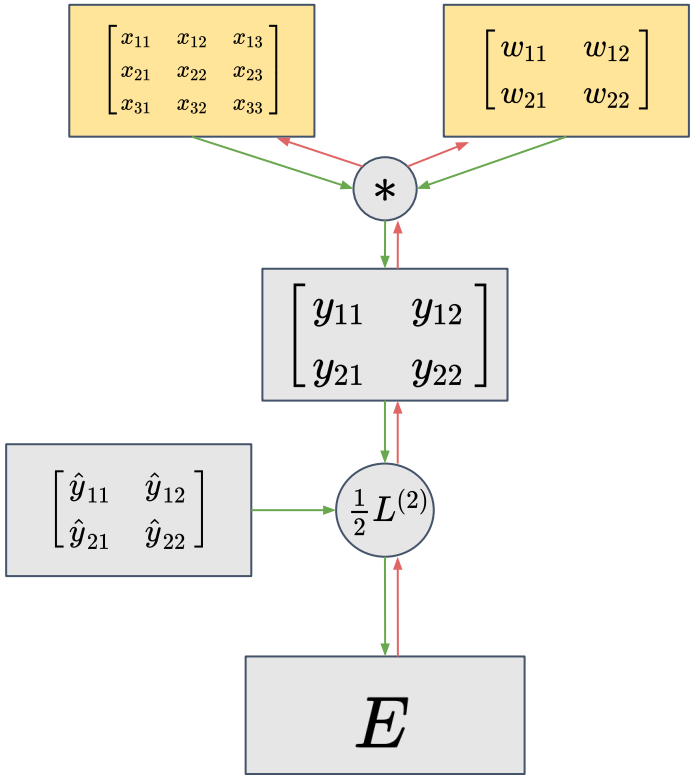
Jeremy Pinto
Applied Research Scientist, Mila
jeremy.pinto@mila.quebec

Warning to .pdf readers

Some slides contained in this presentation contain .gif animations. Pdf files cannot render .gifs so this explains some blank slides in the .pdf format.

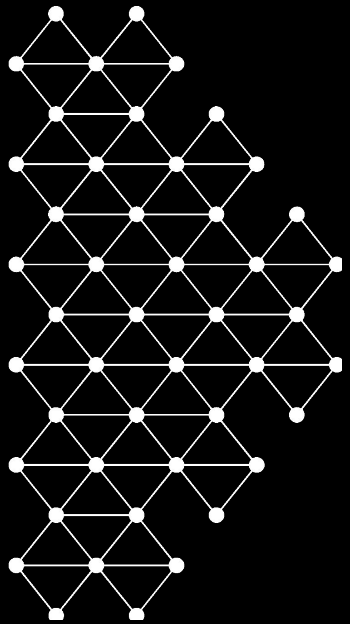
Contents of Part II

- Gradient Descent in CNNs
- Applications of CNNs
- GANs
- Datasets



Quebec
Artificial
Intelligence
Institute

Mila



Gradient Descent in CNNs

WARNING: MATH INCOMING

There is a lot of **math** in the coming slides but do not fear! You should try to follow the main ideas, but should also go back and review them at home with lots of coffee. Don’t be afraid to drink and derive!

https://simple.m.wikipedia.org/wiki/File:Ambox_warning_pn.svg

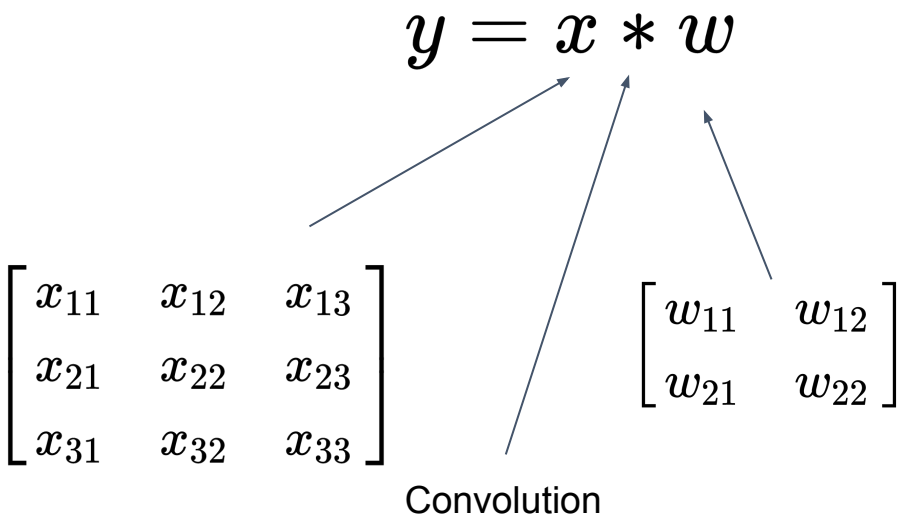
Gradient Descent Recall

As you have seen in Day 2's back propagation lecture, there are multiple steps involved in performing (stochastic) gradient descent:

1. Compute the forward pass
2. Compute the derivative of the loss through a backward pass
3. Update the tunable parameters of the network

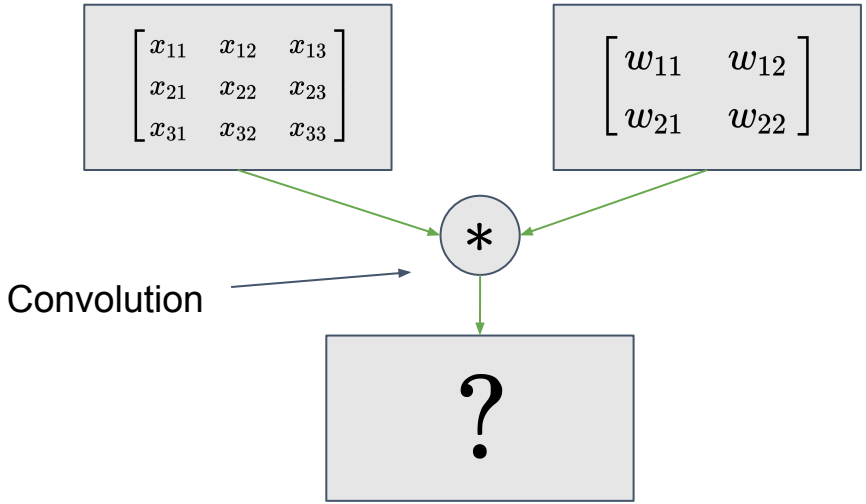
Problem Setup

Consider a simple CNN, where our input, *x*, is 3x3 and kernel, *w,* is 2x2. No padding, stride = 1:

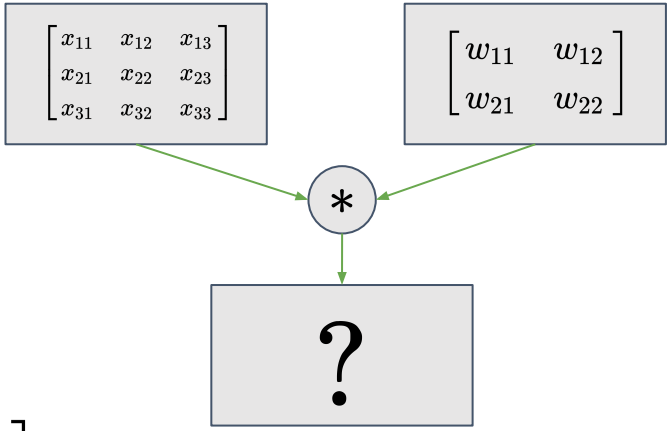


Forward Pass

Graphically, this looks like this:



Forward Pass



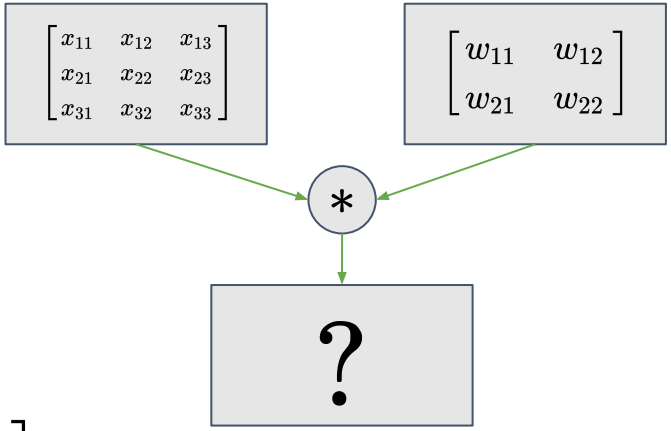
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} \end{bmatrix}$$

Forward Pass



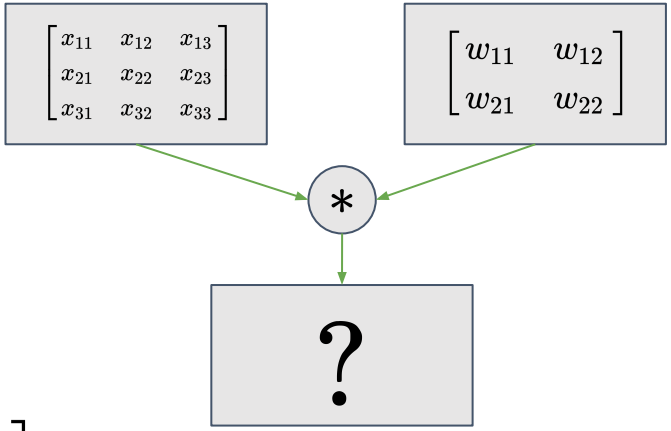
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} \end{bmatrix}$$

Forward Pass



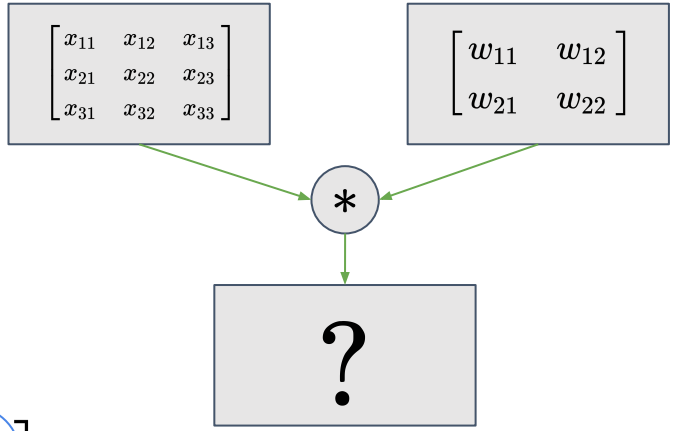
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, \end{bmatrix}$$

Forward Pass



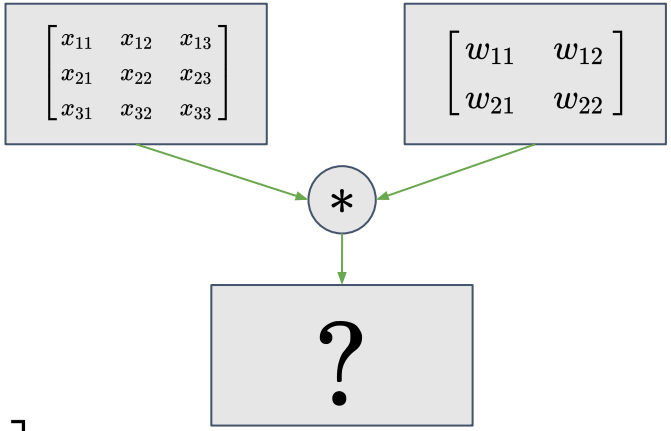
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, \end{bmatrix}$$

Forward Pass



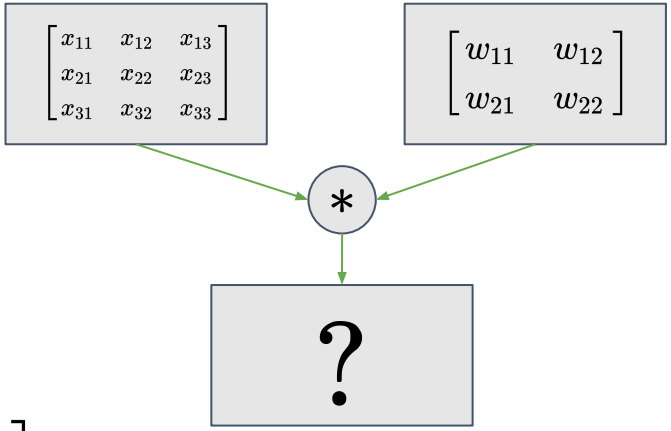
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, \end{bmatrix}$$

Forward Pass



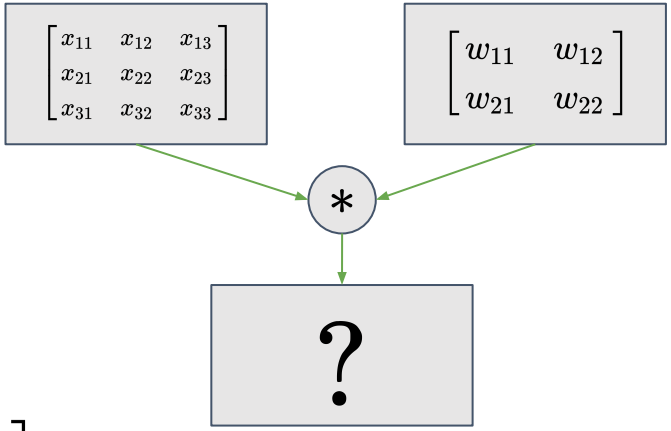
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, \end{bmatrix}$$

Forward Pass



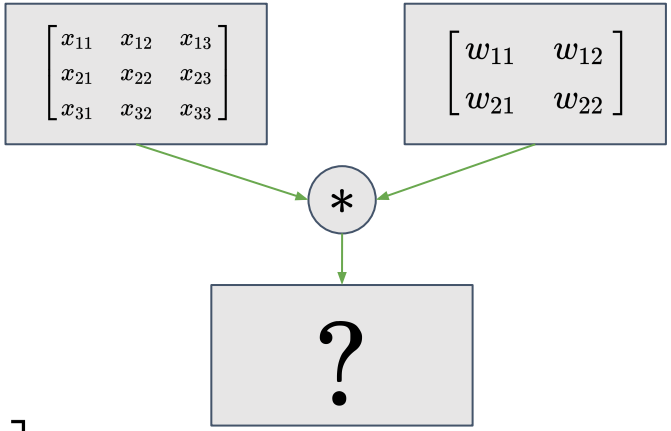
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \end{bmatrix}$$

Forward Pass



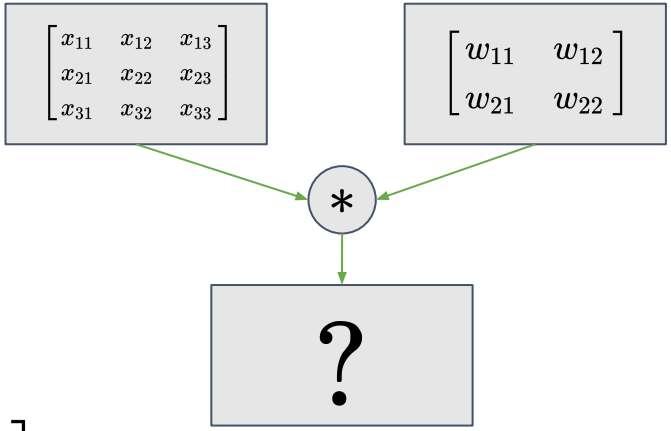
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & \end{bmatrix}$$

Forward Pass



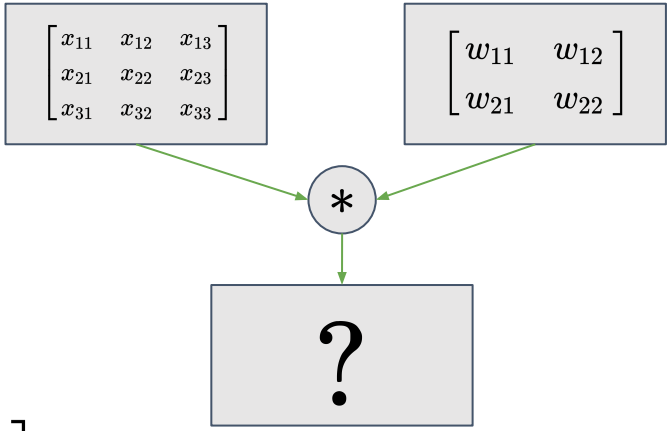
We can compute the forward pass, or the output, *y*:

$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

Forward Pass



We can compute the forward pass, or the output, *y*:

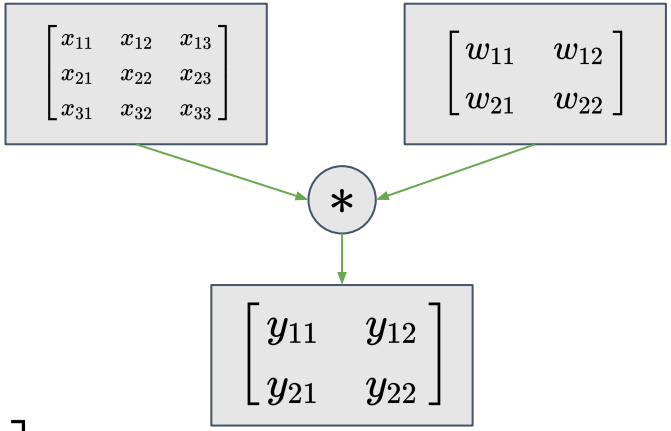
$$y = x * w$$

$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$$

Forward Pass



We can compute the forward pass, or the output, *y*:

$$y = x * w$$

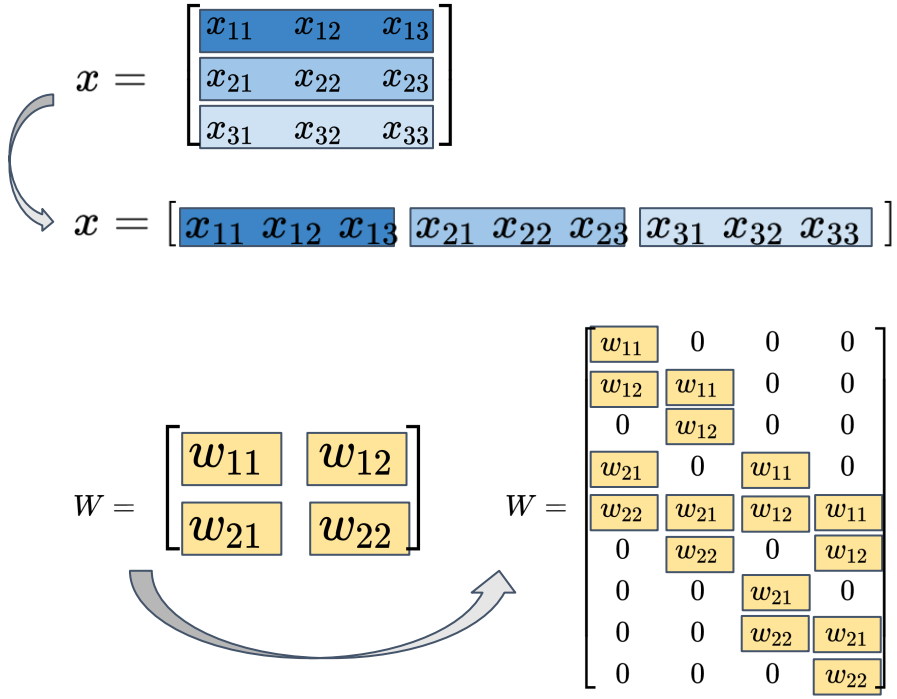
$$= \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$$

Forward Pass

Notice that this is **equivalent** to computing the following **matrix multiplication**: unroll the vector x_{ij} from left to right, top to bottom, construct W accordingly and unroll the resulting vector back to a 2x2 matrix. We will come back to this.



Loss

Now let's define a **loss** function i.e. how we evaluate the **error** between the output we expect, $\hat{y}$, and the output we computed y. In this example, we use **$L^{(2)}$ norm**:

$$E = \frac{1}{2}\sum_{i,j}(y_{ij} - \hat{y}_{ij})^2 = \frac{1}{2}L^{(2)}$$

Notice that the derivative w.r.t. each output can be expressed as:

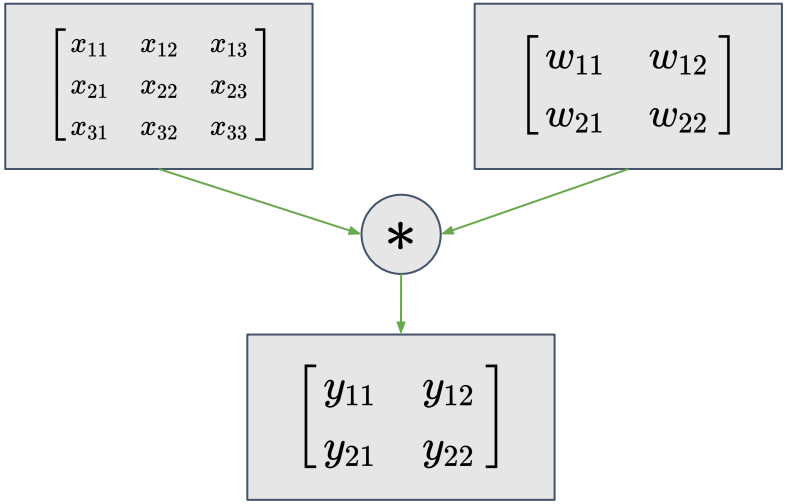
$$\frac{\partial E}{\partial y_{ij}} = (y_{ij} - \hat{y}_{ij})$$

There are many other types of loss functions that we could use, such as Binary Cross Entropy, etc.

Loss

We can continue building our computational graph:

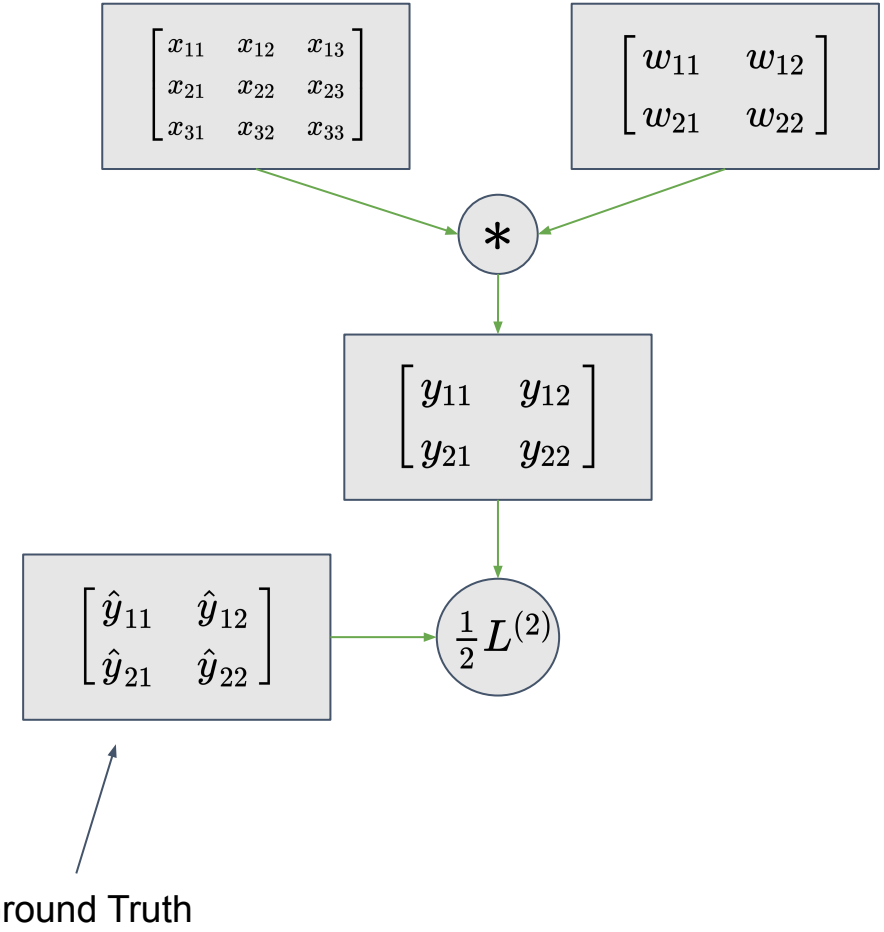
We compute the error between the ground truth and the output predicted by the network.



Loss

We can continue building our computational graph:

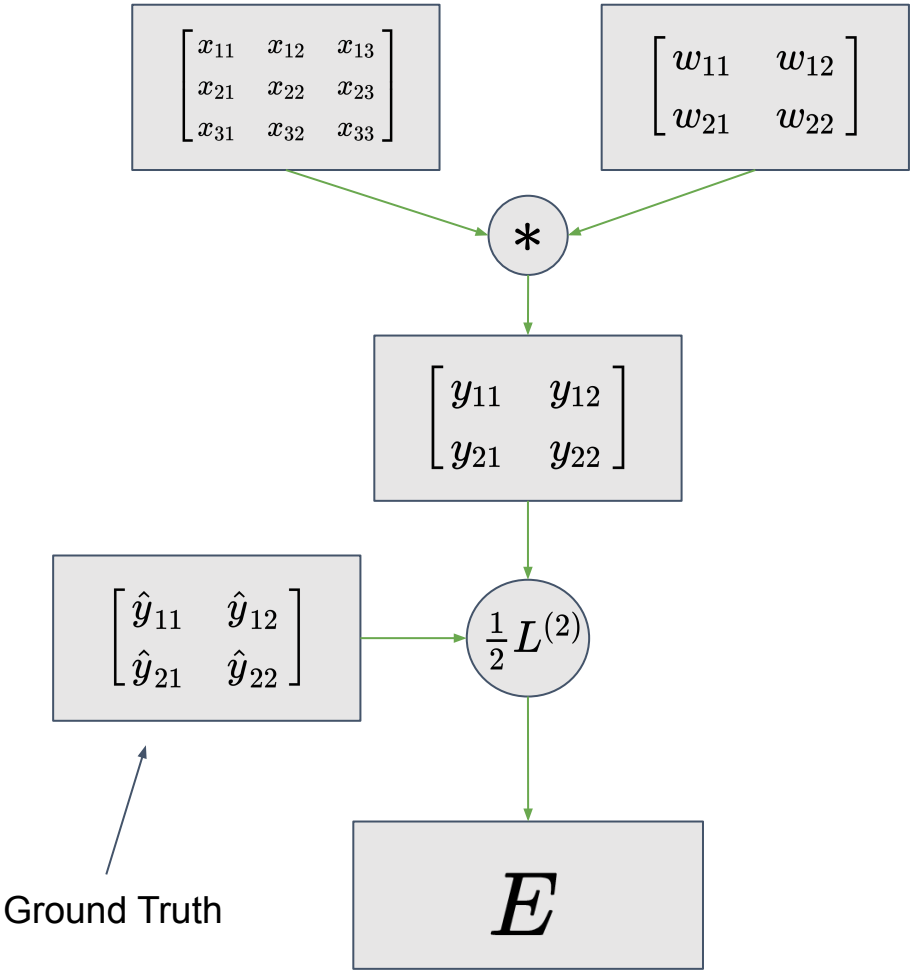
We compute the error between the ground truth and the output predicted by the network.



Loss

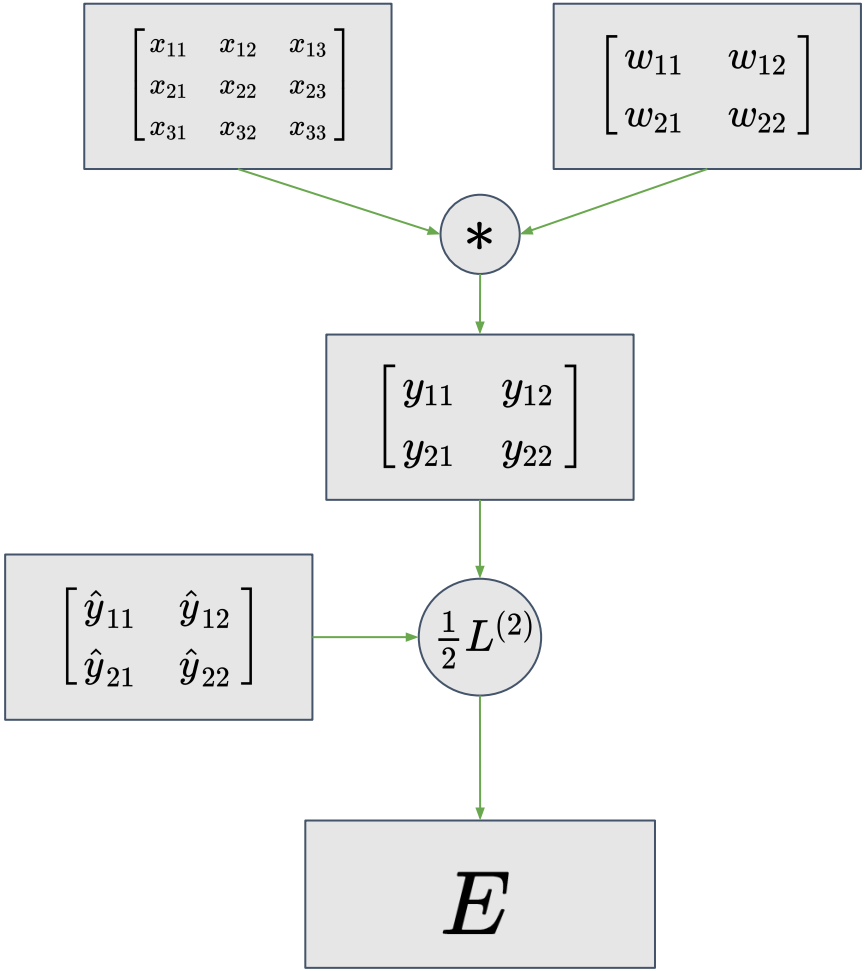
We can continue building our computational graph:

We compute the error between the ground truth and the output predicted by the network.



Gradient Descent Recall

Recall that we are interested in calculating the **update rule** for the tunable parameters.

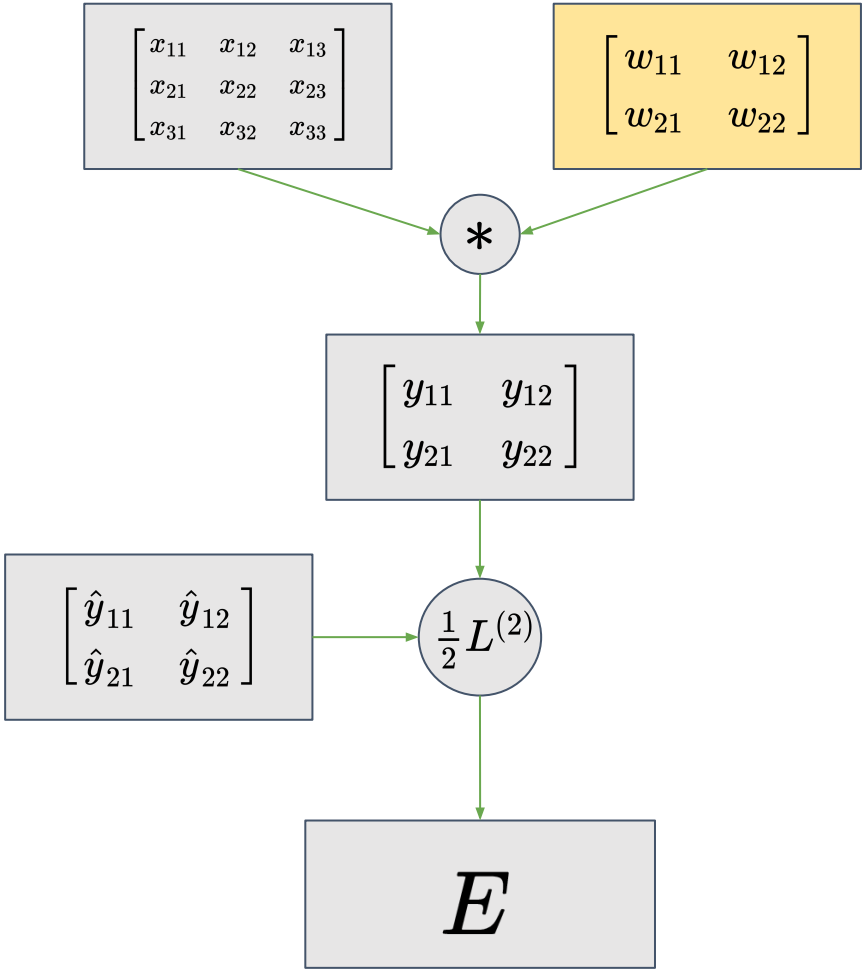


Gradient Descent Recall

Recall that we are interested in calculating the **update rule** for the tunable parameters. The update rule for those parameters are:

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

In this case, the tunable parameters are in yellow.

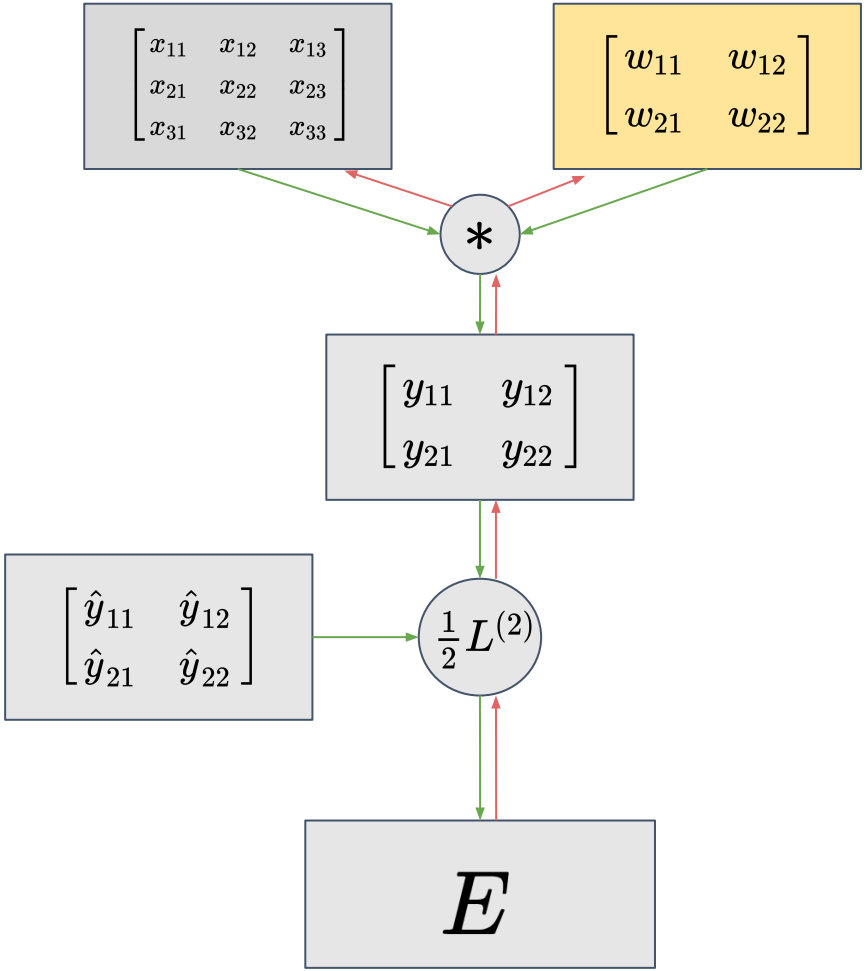


Gradient Descent Recall

Recall that we are interested in calculating the **update rule** for the tunable parameters. The update rule for those parameters are:

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

In this case, the tunable parameters are in yellow. We must therefore calculate the derivative of the error with respect to all the w_{ij}.



Gradient

Let's begin by calculating all the $\frac{\partial E}{\partial w_{ij}}$:

Recall that our output, y, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

$$= \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$$

Since each weight of the kernel affects each output and contributes to the **loss** E, we need to compute the **total derivative** over the output for each kernel weight w_{ij}:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{ij}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{ij}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{ij}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{ij}}$$

Gradient

In our example, we can expand all cases for all *i,j*:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

$$\frac{\partial E}{\partial w_{12}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{12}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{12}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{12}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{12}}$$

$$\frac{\partial E}{\partial w_{21}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{21}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{21}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{21}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{21}}$$

$$\frac{\partial E}{\partial w_{22}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{22}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{22}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{22}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{22}}$$

Gradient

In our example, we can expand all cases for all *i,j*. Now let's focus on *i,j = 1,1*

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

$$\frac{\partial E}{\partial w_{12}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{12}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{12}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{12}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{12}}$$

$$\frac{\partial E}{\partial w_{21}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{21}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{21}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{21}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{21}}$$

$$\frac{\partial E}{\partial w_{22}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{22}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{22}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{22}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{22}}$$

Gradient

In our example:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

Gradient

In our example:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

Let's first compute:

$$\frac{\partial y_{11}}{\partial w_{11}} =$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

Gradient

In our example:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

Let's first compute:

$$\frac{\partial y_{11}}{\partial w_{11}} =$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

y_{11}

Gradient

In our example:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

Let's first compute:

$$\frac{\partial y_{11}}{\partial w_{11}} = x_{11}$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

y_{11}

Gradient

Expanding:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

$$= \frac{\partial E}{\partial y_{11}}x_{11} + \frac{\partial E}{\partial y_{12}}x_{12} + \frac{\partial E}{\partial y_{21}}x_{21} + \frac{\partial E}{\partial y_{22}}x_{22}$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

Gradient

We now have the derivative of the error with respect to the kernel weights

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial w_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial w_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial w_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial w_{11}}$$

$$= \frac{\partial E}{\partial y_{11}}x_{11} + \frac{\partial E}{\partial y_{12}}x_{12} + \frac{\partial E}{\partial y_{21}}x_{21} + \frac{\partial E}{\partial y_{22}}x_{22}$$

These terms represent the error which is computed numerically

$$\frac{\partial E}{\partial y_{ij}} = (y_{ij} - \hat{y}_{ij})$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

Gradient

Expanding for every w_{ij}:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_{11}} x_{11} + \frac{\partial E}{\partial y_{12}} x_{12} + \frac{\partial E}{\partial y_{21}} x_{21} + \frac{\partial E}{\partial y_{22}} x_{22}$$

$$\frac{\partial E}{\partial w_{12}} = \frac{\partial E}{\partial y_{11}} x_{12} + \frac{\partial E}{\partial y_{12}} x_{13} + \frac{\partial E}{\partial y_{21}} x_{22} + \frac{\partial E}{\partial y_{22}} x_{23}$$

$$\frac{\partial E}{\partial w_{21}} = \frac{\partial E}{\partial y_{11}} x_{21} + \frac{\partial E}{\partial y_{12}} x_{22} + \frac{\partial E}{\partial y_{21}} x_{31} + \frac{\partial E}{\partial y_{22}} x_{32}$$

$$\frac{\partial E}{\partial w_{22}} = \frac{\partial E}{\partial y_{11}} x_{22} + \frac{\partial E}{\partial y_{12}} x_{23} + \frac{\partial E}{\partial y_{21}} x_{32} + \frac{\partial E}{\partial y_{22}} x_{33}$$

Gradient

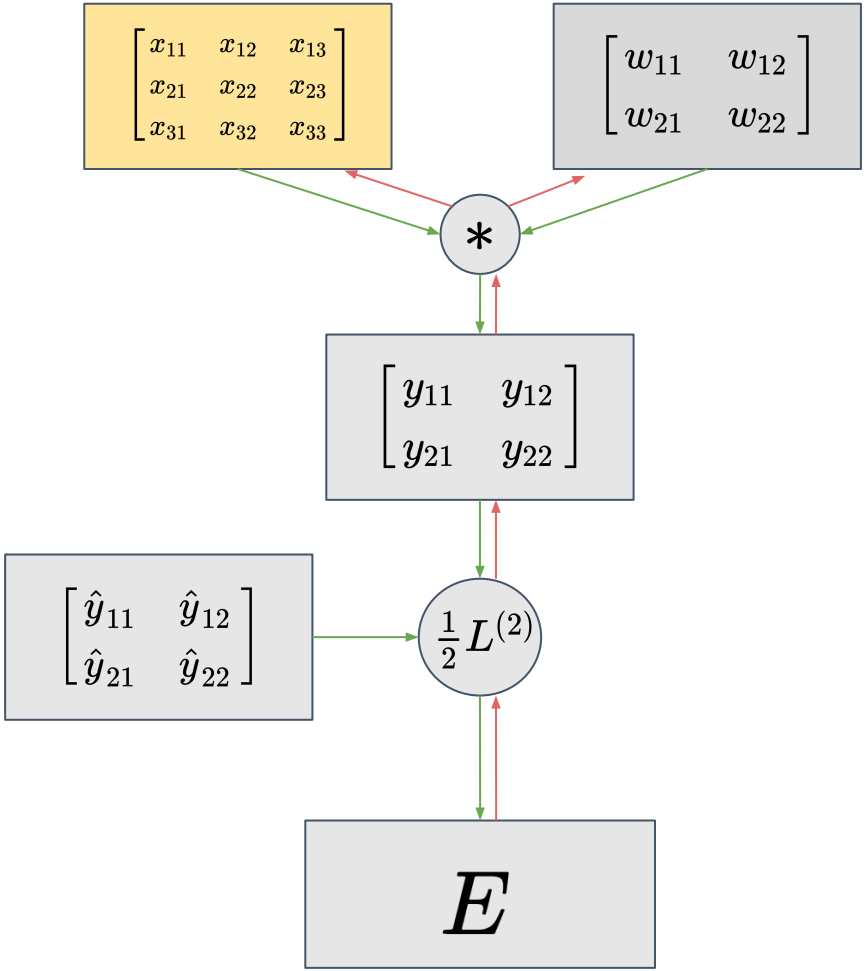
… Which happens to be equivalent to:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \frac{\partial E}{\partial w_{12}} \\ \frac{\partial E}{\partial w_{21}} & \frac{\partial E}{\partial w_{22}} \end{bmatrix}$$

Convolution

Gradient Descent Recall

The inputs x_{ij} can also be the output from a previous layer. We therefore need to compute the derivative with respect to our inputs $\frac{\partial E}{\partial x_{ij}}$.



Gradient

Now, we need to calculate all the $\frac{\partial E}{\partial x_{ij}}$

We need to compute the derivative of the error with respect to the inputs x_{ij}.
In general, for any pair (i,j) we have:

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{ij}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial x_{ij}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial x_{ij}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial x_{ij}}$$

Gradient

Now, we need to calculate all the $\frac{\partial E}{\partial x_{ij}}$

We need to compute the derivative of the error with respect to the inputs x_{ij}.
In general, for any pair (i,j) we have:

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{ij}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial x_{ij}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial x_{ij}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial x_{ij}}$$

Here expand for (i,j) = (1,1)

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial x_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial x_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial x_{11}}$$

Gradient

For (i,j) = (1,1)

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}} \frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}} \frac{\partial y_{12}}{\partial x_{11}} + \frac{\partial E}{\partial y_{21}} \frac{\partial y_{21}}{\partial x_{11}} + \frac{\partial E}{\partial y_{22}} \frac{\partial y_{22}}{\partial x_{11}}$$

Gradient

For (i,j) = (1,1)

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial x_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial x_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial x_{11}}$$

Let's first compute:

$$\frac{\partial y_{11}}{\partial x_{11}} =$$

Gradient

For (i,j) = (1,1)

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial x_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial x_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial x_{11}}$$

Let's first compute:

$$\frac{\partial y_{11}}{\partial x_{11}} =$$

Recall that our output, *y*, is:

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

y_{11}

Gradient

For (i,j) = (1,1)

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{11}} + \frac{\partial E}{\partial y_{12}}\frac{\partial y_{12}}{\partial x_{11}} + \frac{\partial E}{\partial y_{21}}\frac{\partial y_{21}}{\partial x_{11}} + \frac{\partial E}{\partial y_{22}}\frac{\partial y_{22}}{\partial x_{11}}$$

Let's first compute:

$$\frac{\partial y_{11}}{\partial x_{11}} = w_{11}$$

Recall that our output, *y*, is:

y_{11}

$$y = x * w = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22}, & x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} \\ x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22}, & x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} \end{bmatrix}$$

Gradient

Expanding for all (i,j), we get:

$$
\begin{aligned}
\frac{\partial E}{\partial x_{11}} &= \frac{\partial E}{\partial y_{11}} w_{11} \\
\frac{\partial E}{\partial x_{12}} &= \frac{\partial E}{\partial y_{11}} w_{12} + \frac{\partial E}{\partial y_{12}} w_{11} \\
\frac{\partial E}{\partial x_{13}} &= \frac{\partial E}{\partial y_{12}} w_{12} \\
\frac{\partial E}{\partial x_{21}} &= \frac{\partial E}{\partial y_{11}} w_{21} + \frac{\partial E}{\partial y_{21}} w_{11} \\
\frac{\partial E}{\partial x_{22}} &= \frac{\partial E}{\partial y_{11}} w_{22} + \frac{\partial E}{\partial y_{12}} w_{21} + \frac{\partial E}{\partial y_{21}} w_{12} + \frac{\partial E}{\partial y_{22}} w_{11} \\
\frac{\partial E}{\partial x_{23}} &= \frac{\partial E}{\partial y_{12}} w_{22} + \frac{\partial E}{\partial y_{22}} w_{12} \\
\frac{\partial E}{\partial x_{31}} &= \frac{\partial E}{\partial y_{21}} w_{21} \\
\frac{\partial E}{\partial x_{32}} &= \frac{\partial E}{\partial y_{21}} w_{22} + \frac{\partial E}{\partial y_{22}} w_{21} \\
\frac{\partial E}{\partial x_{33}} &= \frac{\partial E}{\partial y_{22}} w_{22}
\end{aligned}
$$

Notice the **symmetry**? This is equivalent to calculating and reshaping the following matrix multiplication:

$$
\partial E y = \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} & \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix}
$$

$$
W^T = \begin{bmatrix}
w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\
0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\
0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22}
\end{bmatrix}
$$

$$
\partial E y \times W^T =
$$

Gradient

We are using the same **kernel matrix** we used to compute the forward pass. This is called "**Transposed Convolution**"

$$\partial Ey = \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} & \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix}$$

$$W^T = \begin{bmatrix} w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix}$$

$$\partial Ey \times W^T =$$

$$\begin{aligned}
\frac{\partial E}{\partial x_{11}} &= \frac{\partial E}{\partial y_{11}} w_{11} \\
\frac{\partial E}{\partial x_{12}} &= \frac{\partial E}{\partial y_{11}} w_{12} + \frac{\partial E}{\partial y_{12}} w_{11} \\
\frac{\partial E}{\partial x_{13}} &= \frac{\partial E}{\partial y_{12}} w_{12} \\
\frac{\partial E}{\partial x_{21}} &= \frac{\partial E}{\partial y_{11}} w_{21} + \frac{\partial E}{\partial y_{21}} w_{11} \\
\frac{\partial E}{\partial x_{22}} &= \frac{\partial E}{\partial y_{11}} w_{22} + \frac{\partial E}{\partial y_{12}} w_{21} + \frac{\partial E}{\partial y_{21}} w_{12} + \frac{\partial E}{\partial y_{22}} w_{11} \\
\frac{\partial E}{\partial x_{23}} &= \frac{\partial E}{\partial y_{12}} w_{22} + \frac{\partial E}{\partial y_{22}} w_{12} \\
\frac{\partial E}{\partial x_{31}} &= \frac{\partial E}{\partial y_{21}} w_{21} \\
\frac{\partial E}{\partial x_{32}} &= \frac{\partial E}{\partial y_{21}} w_{22} + \frac{\partial E}{\partial y_{22}} w_{21} \\
\frac{\partial E}{\partial x_{33}} &= \frac{\partial E}{\partial y_{22}} w_{22}
\end{aligned}$$

Putting it all together

The **forward pass** can be computed via matrix multiplication:

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ 0 & w_{12} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ 0 & w_{22} & 0 & w_{12} \\ 0 & 0 & w_{21} & 0 \\ 0 & 0 & w_{22} & w_{21} \\ 0 & 0 & 0 & w_{22} \end{bmatrix}$$

Putting it all together

The **forward pass** can be computed via matrix multiplication:

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ 0 & w_{12} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ 0 & w_{22} & 0 & w_{12} \\ 0 & 0 & w_{21} & 0 \\ 0 & 0 & w_{22} & w_{21} \\ 0 & 0 & 0 & w_{22} \end{bmatrix}$$

The **gradient** w.r.t. the **kernel** weights can be calculated via **convolution**:

$$\begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \frac{\partial E}{\partial w_{12}} \\ \frac{\partial E}{\partial w_{21}} & \frac{\partial E}{\partial w_{22}} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix}$$

Putting it all together

The **forward pass** can be computed via matrix multiplication:

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{bmatrix}$$

$$W = \begin{bmatrix} w_{11} & 0 & 0 & 0 \\ w_{12} & w_{11} & 0 & 0 \\ 0 & w_{12} & 0 & 0 \\ w_{21} & 0 & w_{11} & 0 \\ w_{22} & w_{21} & w_{12} & w_{11} \\ 0 & w_{22} & 0 & w_{12} \\ 0 & 0 & w_{21} & 0 \\ 0 & 0 & w_{22} & w_{21} \\ 0 & 0 & 0 & w_{22} \end{bmatrix}$$

The **gradient** w.r.t. the **kernel** weights can be calculated via **convolution**:

$$\begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \frac{\partial E}{\partial w_{12}} \\ \frac{\partial E}{\partial w_{21}} & \frac{\partial E}{\partial w_{22}} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} \\ \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix}$$

The **gradients** w.r.t. the **inputs** can be computed by "**Transposed convolution**":

$$\frac{\partial E}{\partial x_{ij}} = \partial Ey \times W^T$$

$$\partial Ey = \begin{bmatrix} \frac{\partial E}{\partial y_{11}} & \frac{\partial E}{\partial y_{12}} & \frac{\partial E}{\partial y_{21}} & \frac{\partial E}{\partial y_{22}} \end{bmatrix}$$

$$W^T = \begin{bmatrix} w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix}$$

Ok. Breathe.

Most modern frameworks take care of implementing this for you. You almost never have to look at this math when implementing your own CNN. However, it's important to understand it at least once.

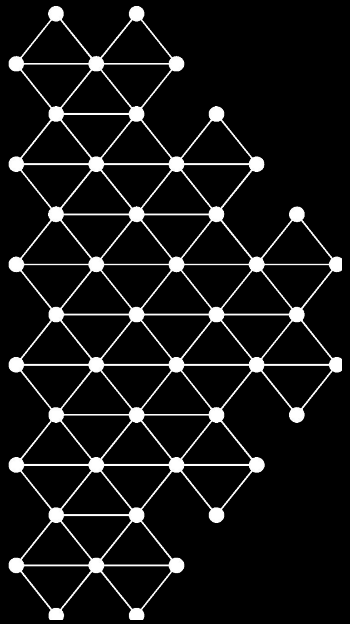
Keep in mind, this was for a single layer, no bias. This was one of the simplest examples we could have derived.

For more details:

https://arxiv.org/pdf/1603.07285.pdf
https://medium.com/@2017csm1006/forward-and-backpropagation-in-convolutional-neural-network-4dfa96d7b37e
https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/
https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

Quebec
Artificial
Intelligence
Institute

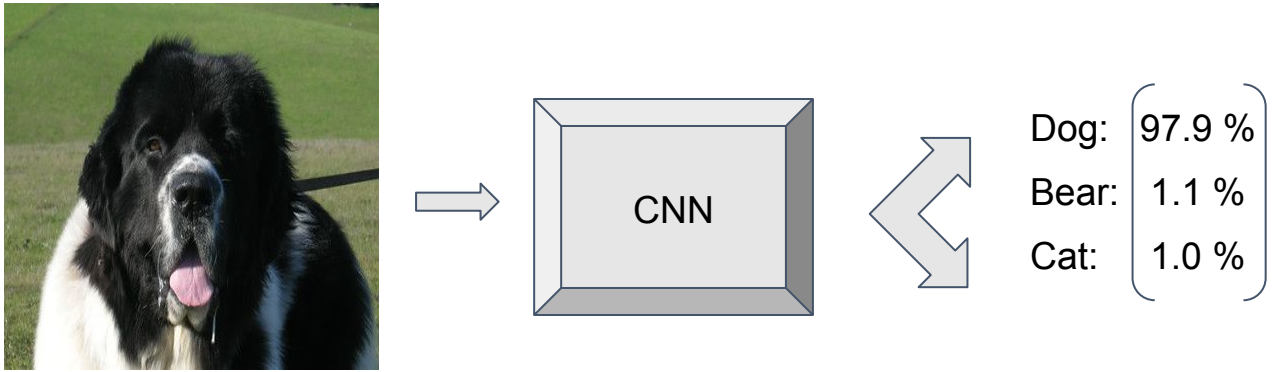
Mila



CNN Applications

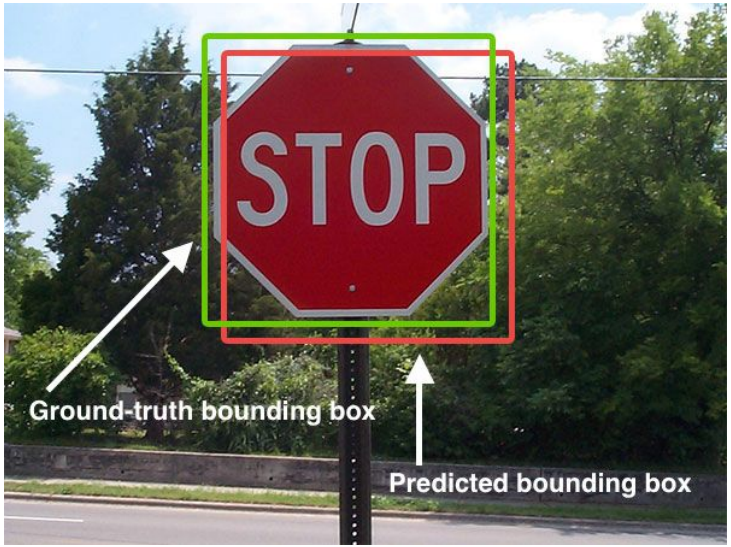
Image Classification

Image classification can be defined as “given an image, what category does the image belong to?”. CNNs are not the only way to do this but happen to be very good at it. This is the most commonly used task for CNNs.



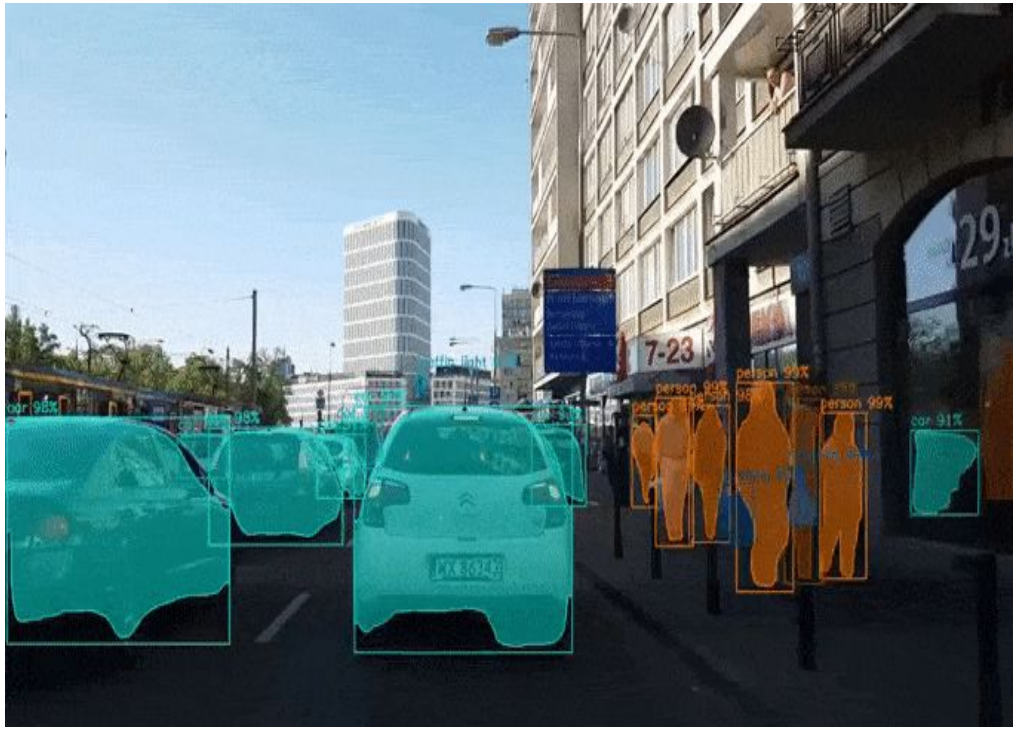
Object Detection

In object detection, coordinates to a bounding box are predicted, as well as the category of object the box belongs to.



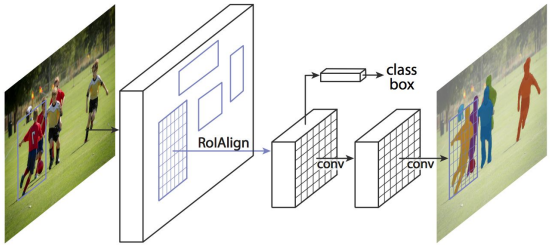
https://upload.wikimedia.org/wikipedia/commons/2/2d/Intersection_over_Union_-_object_detection_bounding_boxes.jpg

Image Segmentation



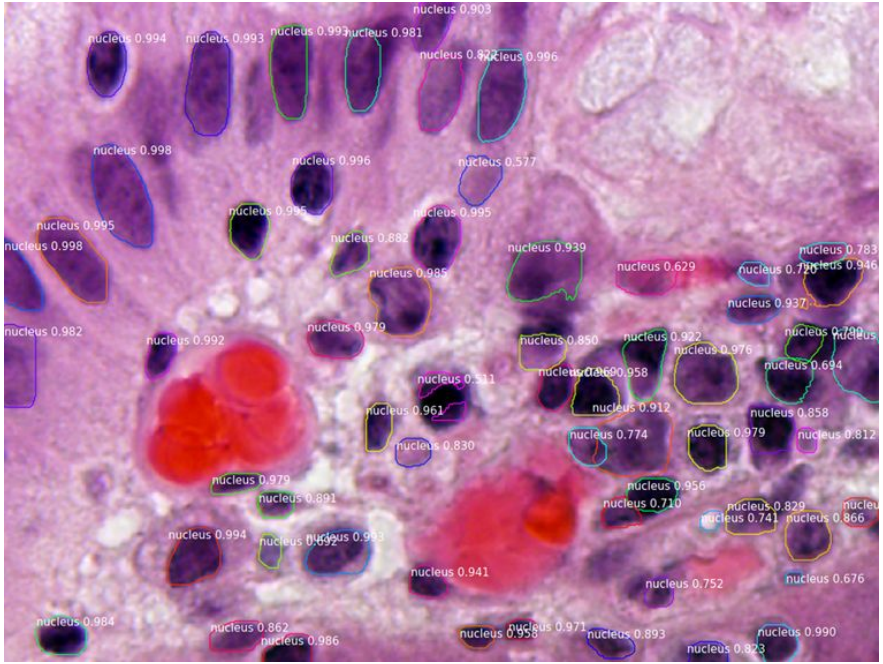
https://github.com/matterport/Mask_RCNN

Image segmentation is the next logical step in object detection; it determines which pixels correspond to the detected objects.



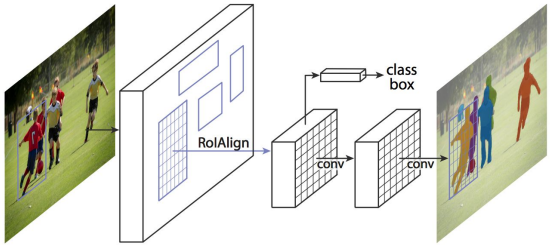
Mask RCNN Architecture

Image Segmentation



https://github.com/matterport/Mask_RCNN

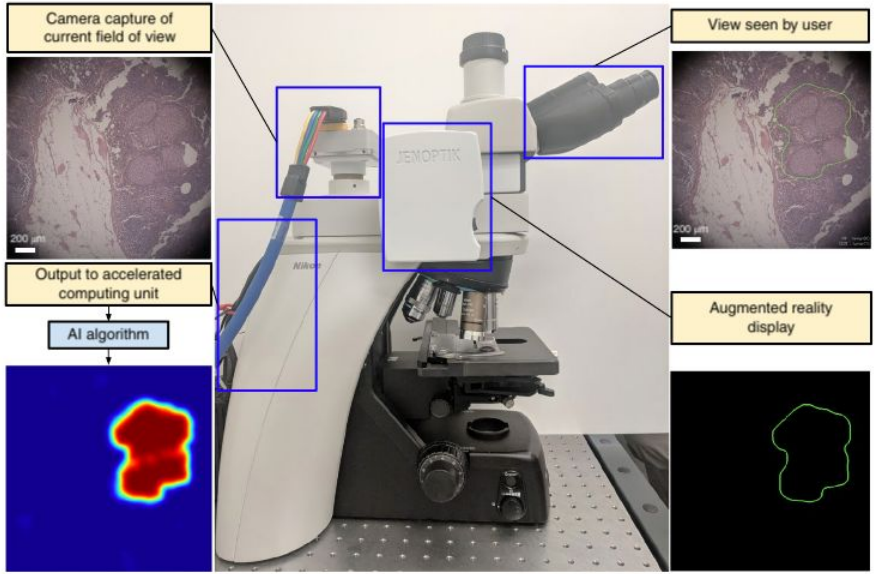
It can be used in a wide variety of fields, such as medical imaging.



Mask RCNN Architecture

Augmented (Medical) Reality

Here is an example of a microscope rendering in real-time regions of interest in medical images.



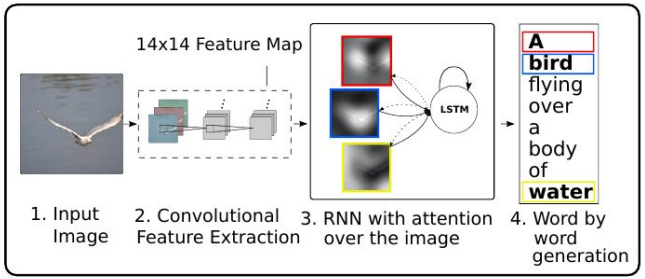
source

Caption Generation

CNNs can be used in a pipeline (along with other NLP models) to help models learn to generate captions.

A woman is throwing a frisbee in a park.

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in Sections 3.1 & 5.4



https://arxiv.org/pdf/1502.03044.pdf

Human Pose Estimation

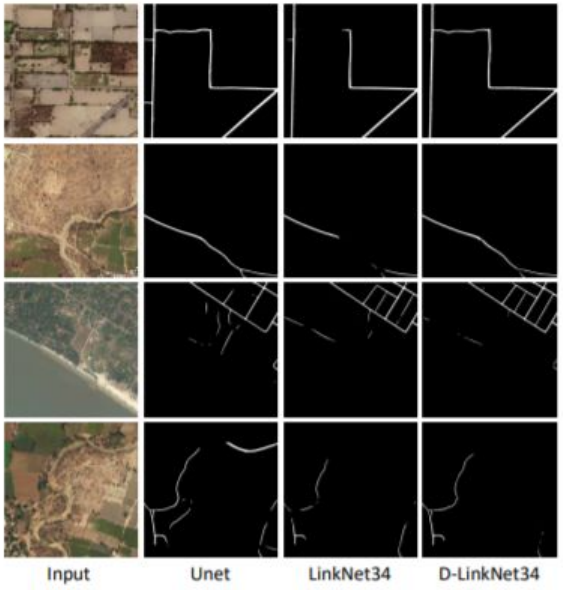
It is possible to use CNNs to extract human poses from images

https://github.com/rachit2403/Open-Pose-Keras

https://github.com/facebookresearch/DensePose

Road Mapping

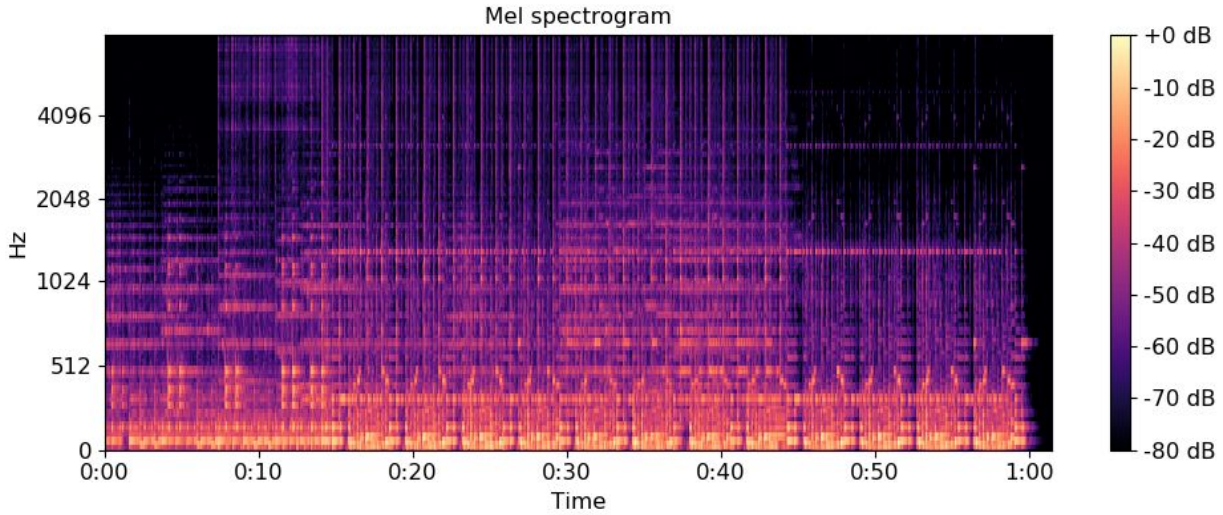
Using satellite imagery, we can automate road + building mapping.



http://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w4/Zhou_D-LinkNet_LinkNet_With_CVPR_2018_paper.pdf
https://ai.facebook.com/blog/mapping-roads-through-deep-learning-and-weakly-supervised-training/
https://github.com/matterport/Mask_RCNN

Sound Classification

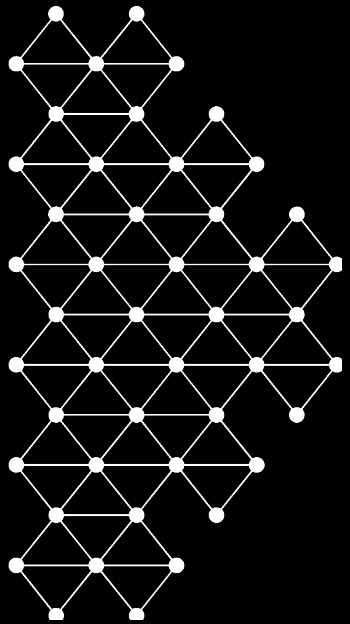
Spectrograms of sounds can be computed using fourier transforms. The spectrograms can then be treated just like images in the context of sound classification.



https://librosa.github.io/librosa/generated/librosa.feature.melspectrogram.html

Quebec
Artificial
Intelligence
Institute

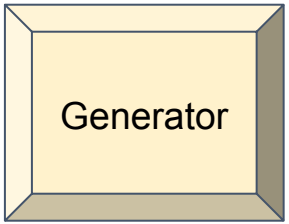
Mila



Generative Adversarial Networks (GANs)

GANs

Generative Adversarial Networks, or GANs, are a special kind of algorithms that make two separate networks compete against each other. The first network is called a **generator** and the other a **discriminator**.



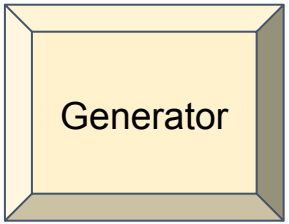
Generates samples that look convincingly real

Determines whether a sample is real or generated

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

GANs

This can be thought of as a game between a **counterfeiter** and an **art curator**. The counterfeiter must make fake images that look so real that the curator will falsely identify them as real.



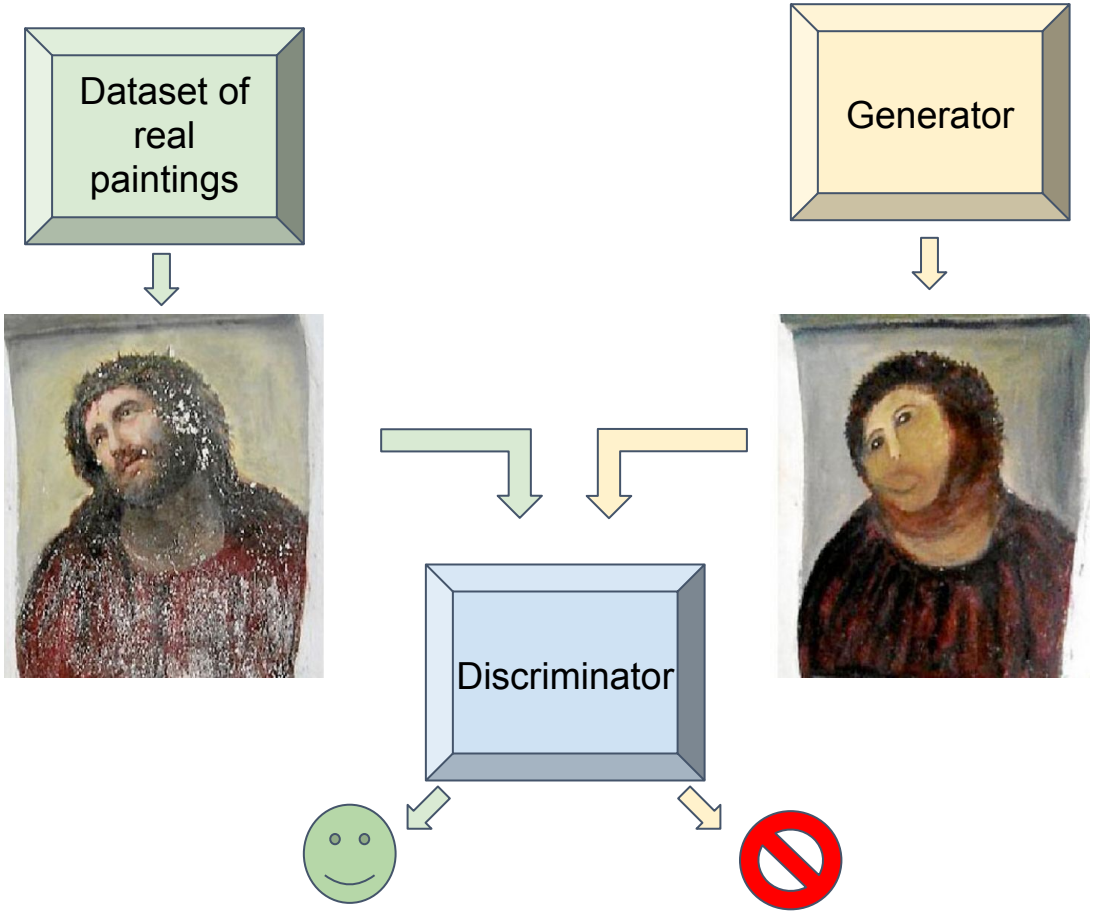
Generates samples that look convincingly real

Determines whether a sample is real or generated

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

GANs

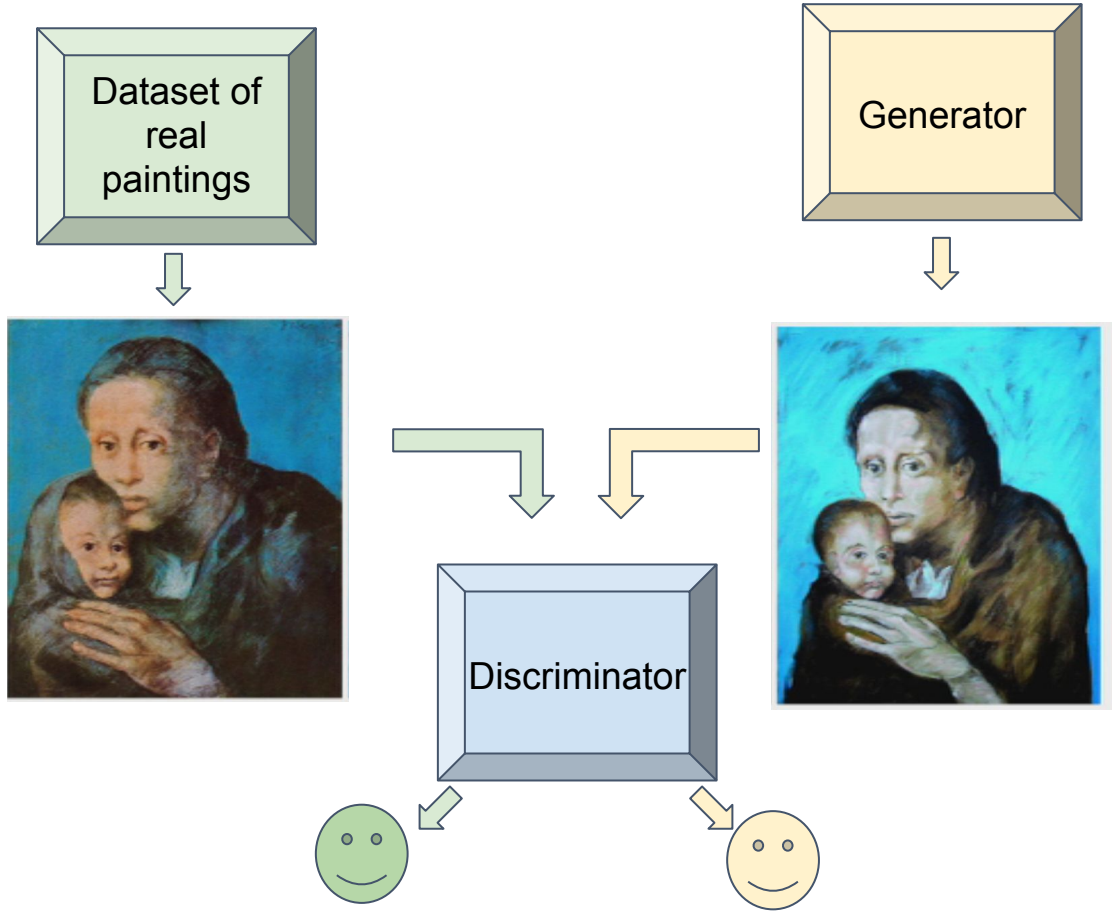
The counterfeiter (generator) attempts to generate real looking images. At first the generator is pretty bad.



https://www.pri.org/stories/2012-08-25/amateur-restoration-botches-jesus-painting-spain

GANs

Eventually, the generator learns to fool the discriminator and can generate convincing images.



https://www.channel4.com/news/art-forgery-beltracchi-wolfgang-ernst-picasso-paraic-obrien

GANs

Mathematically, this can be expressed as a **MinMax** game, where the generator tries to **minimize** the objective function *V* while the discriminator tries to **maximize** it. Both networks are trained successively.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Where:

$D(x)$ is a neural network that predicts how confident it is that the input image $\boldsymbol{x}$ is real. The output of *D* can be interpreted as “the **probability** that the sample is real”.

$G(z)$ is a neural network that generates an image given a noise signal *z*

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

GANs

When training the **generator**, we want to **minimize** the term in blue, i.e. we want to maximize the error that *D* will make on a generated image *G(z).*

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$D(x)$ is a neural network that predicts how confident it is that the input image $\boldsymbol{x}$ is real. The output of *D* can be interpreted as "the **probability** that the sample is real".

$G(z)$ is a neural network that generates an image given a noise signal *z*

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

GANs

When training the **discriminator**, we want to **maximize** the terms in red, i.e. we want to minimize the error that *D* will make on a generated image *G(z).*

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

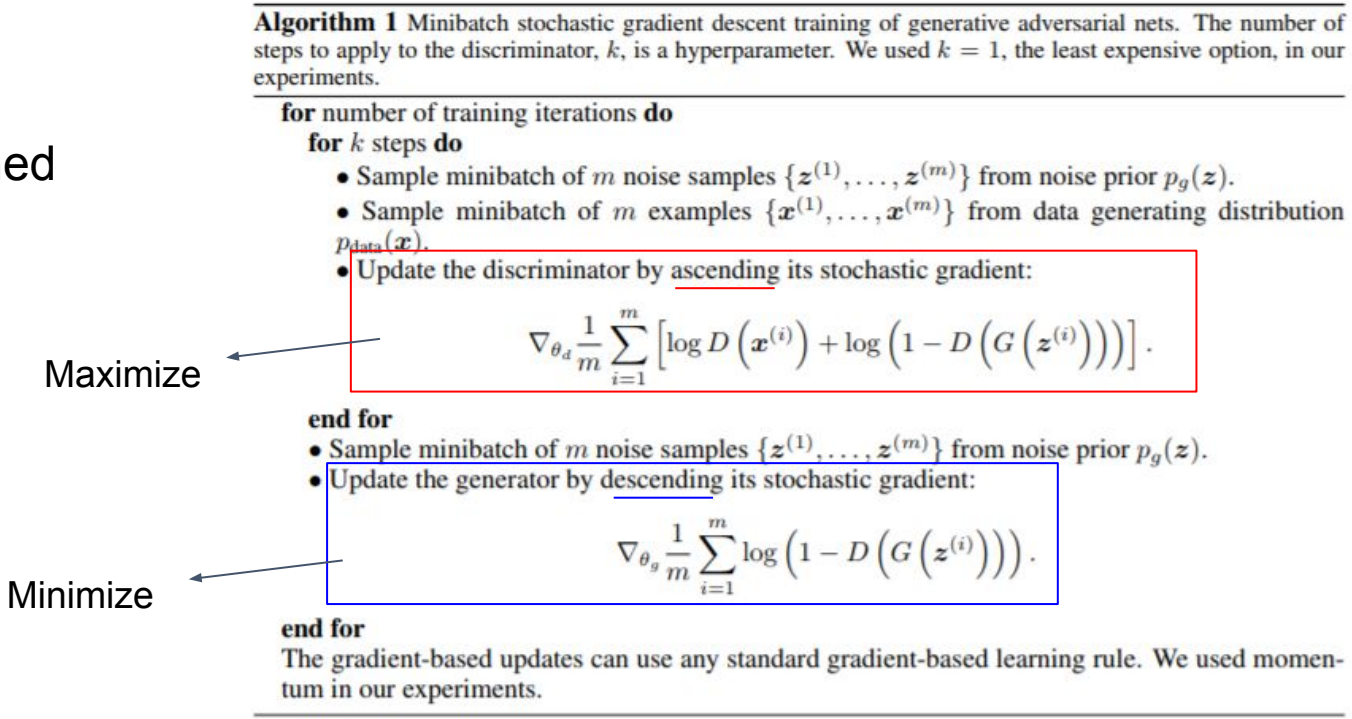
$D(x)$ is a neural network that predicts how confident it is that the input image $\boldsymbol{x}$ is real. The output of *D* can be interpreted as "the **probability** that the sample is real".

$G(z)$ is a neural network that generates an image given a noise signal *z*

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

GANs

Notice that the discriminator and generator are trained **iteratively**.



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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:

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

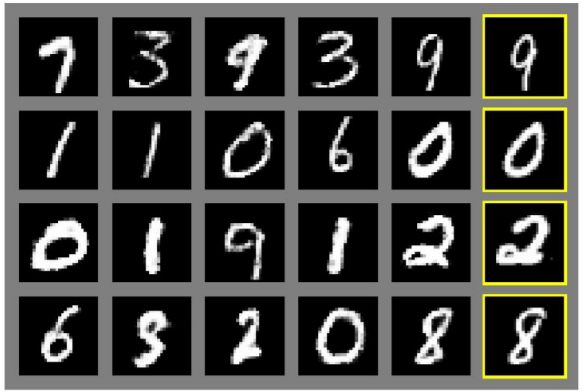
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

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

GANs

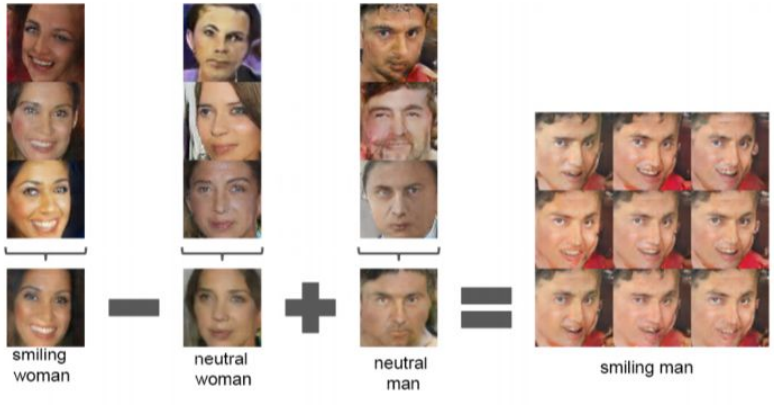
In the original paper, both digits and faces were generated by the netweork and look convincingly real. The yellow boxes show the closest match to its generated neighbors in the training dataset.



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

DCGAN

In a follow up paper, CNNs are used to generate the images. Transposed convolutions are used for upsampling.



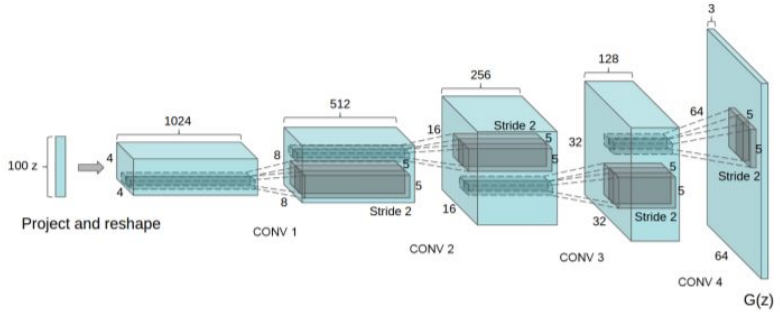


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

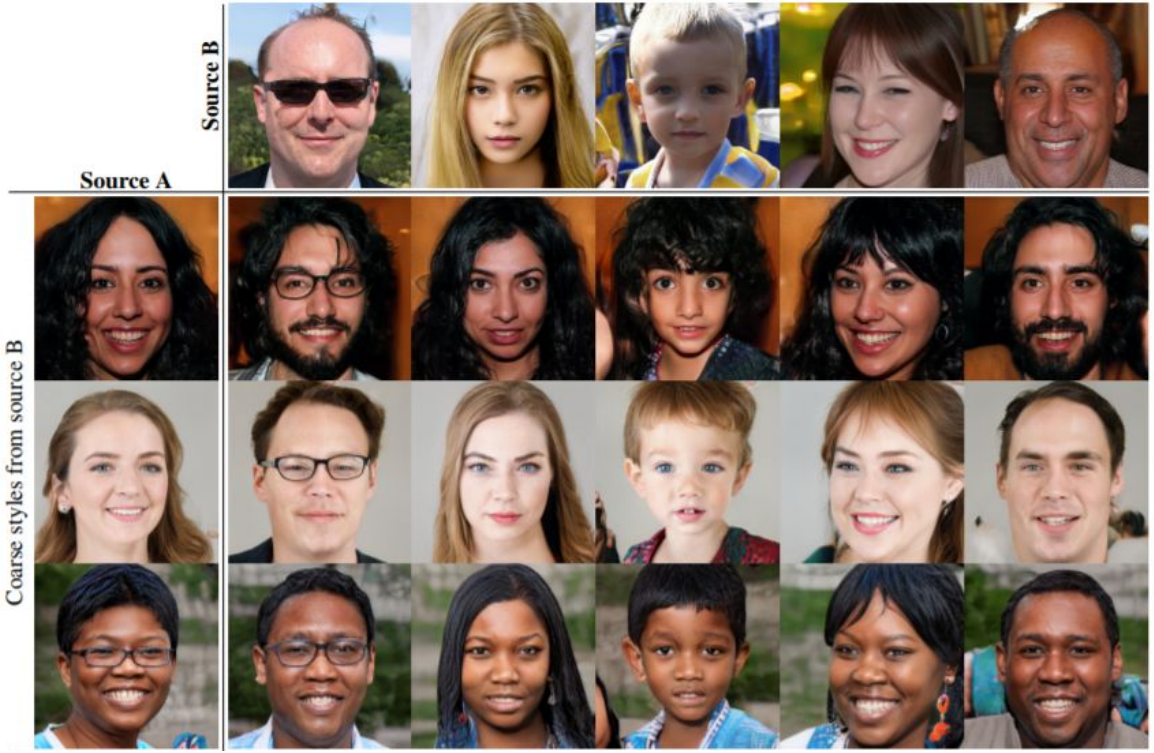
https://arxiv.org/pdf/1511.06434.pdf

Image Synthesis

We can teach networks to generate realistic images that have never been seen before by the network. Which face is fake?

Source

Style Transfer



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

Style Transfer

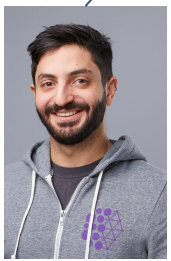
CycleGAN

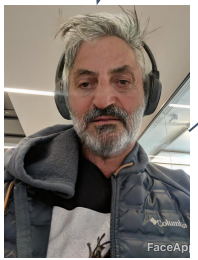
CycleGan introduced the concept of **cycle-consistency** loss to their GANs. This allowed training generators that would unpaired images from domain X to domain Y.

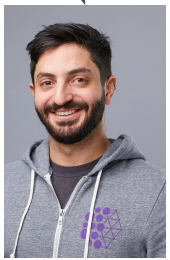
Aging Generator

"*De-Aging*" Generator

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$$







$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

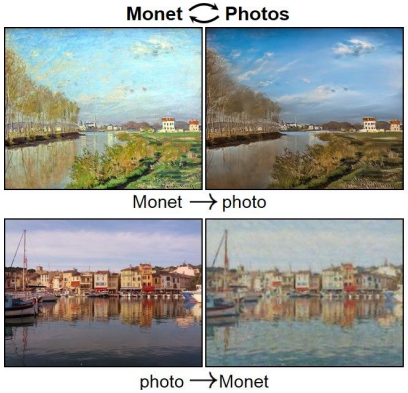
https://arxiv.org/pdf/1703.10593.pdf

https://junyanz.github.io/CycleGAN/

CycleGAN

This allows us to map between arbitrary domains with realistic results.

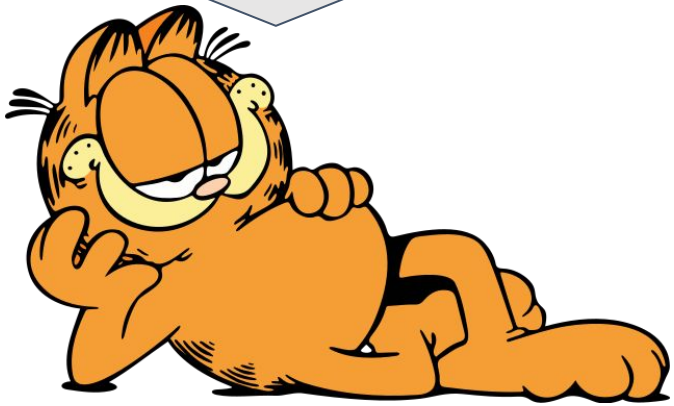
$horse \rightarrow zebra$

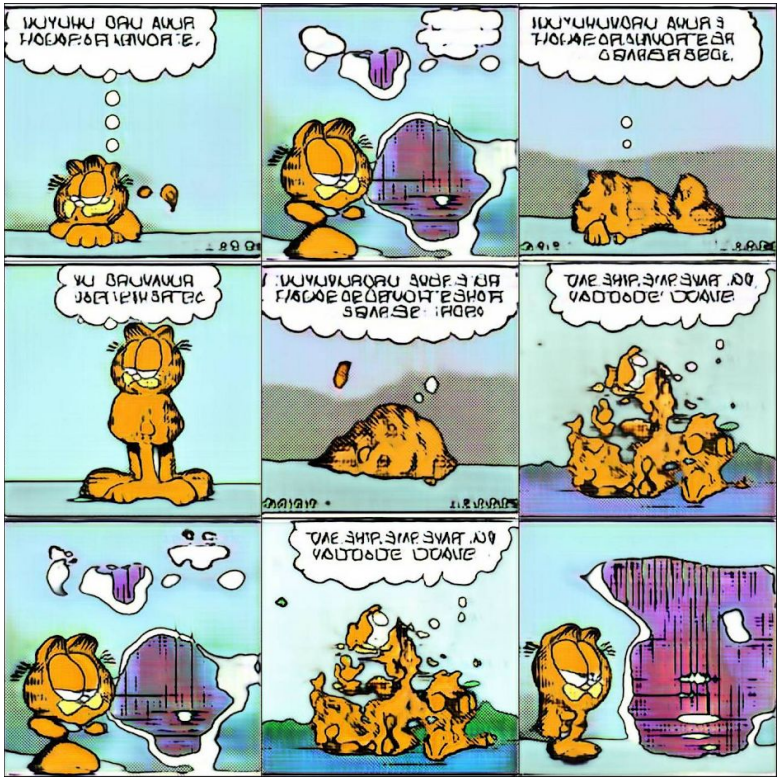


https://junyanz.github.io/CycleGAN/
https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

GANfield

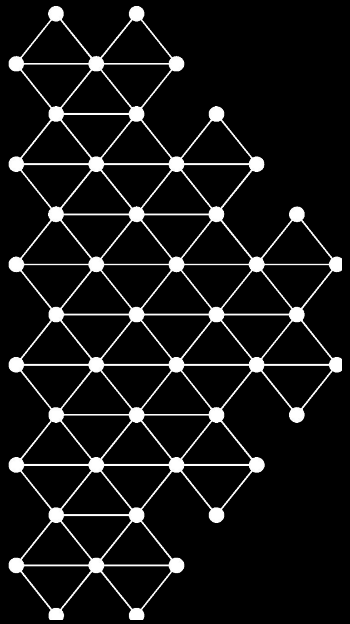
You can use GANs to generate convincingly new and original Garfield content





https://vdalv.github.io/2018/12/04/ganfield.html
https://en.wikipedia.org/wiki/File:Garfield_the_Cat.svg

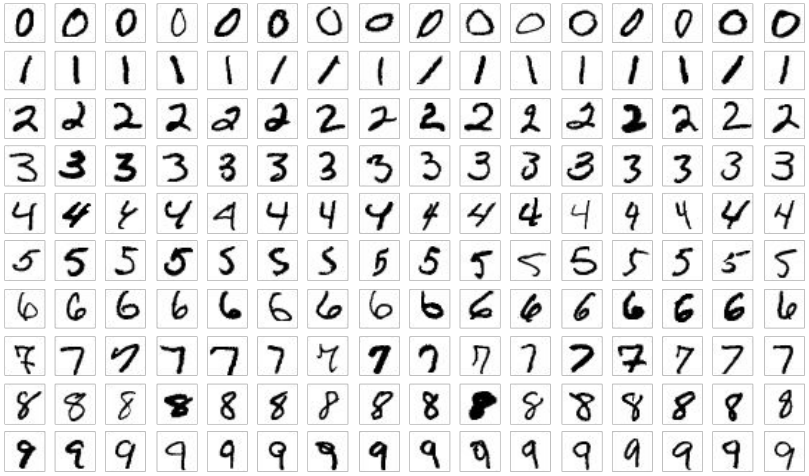
Quebec
Artificial
Intelligence
Institute



CNN Datasets

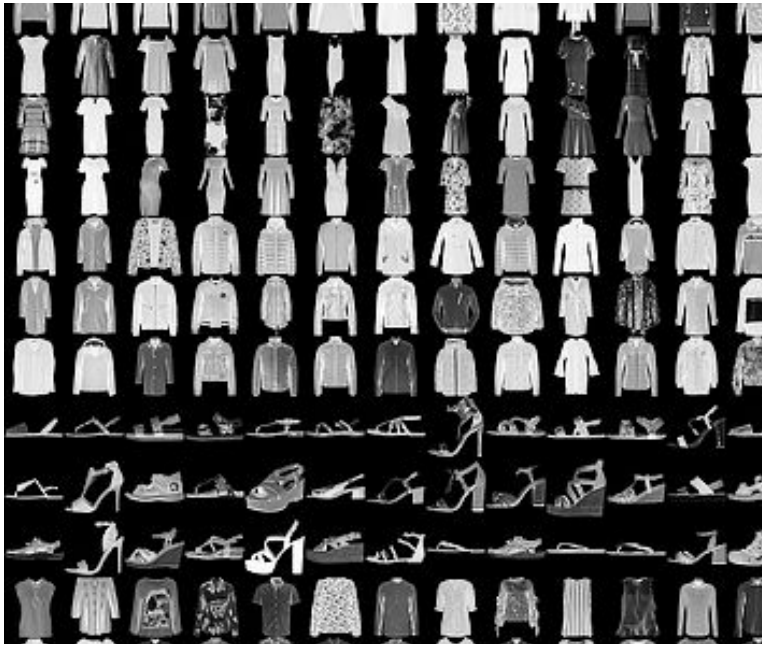
MNIST

- 60 000 handwritten digits
- Grayscale images, 28 x 28 resolution
- Easy to train



Fashion-MNIST

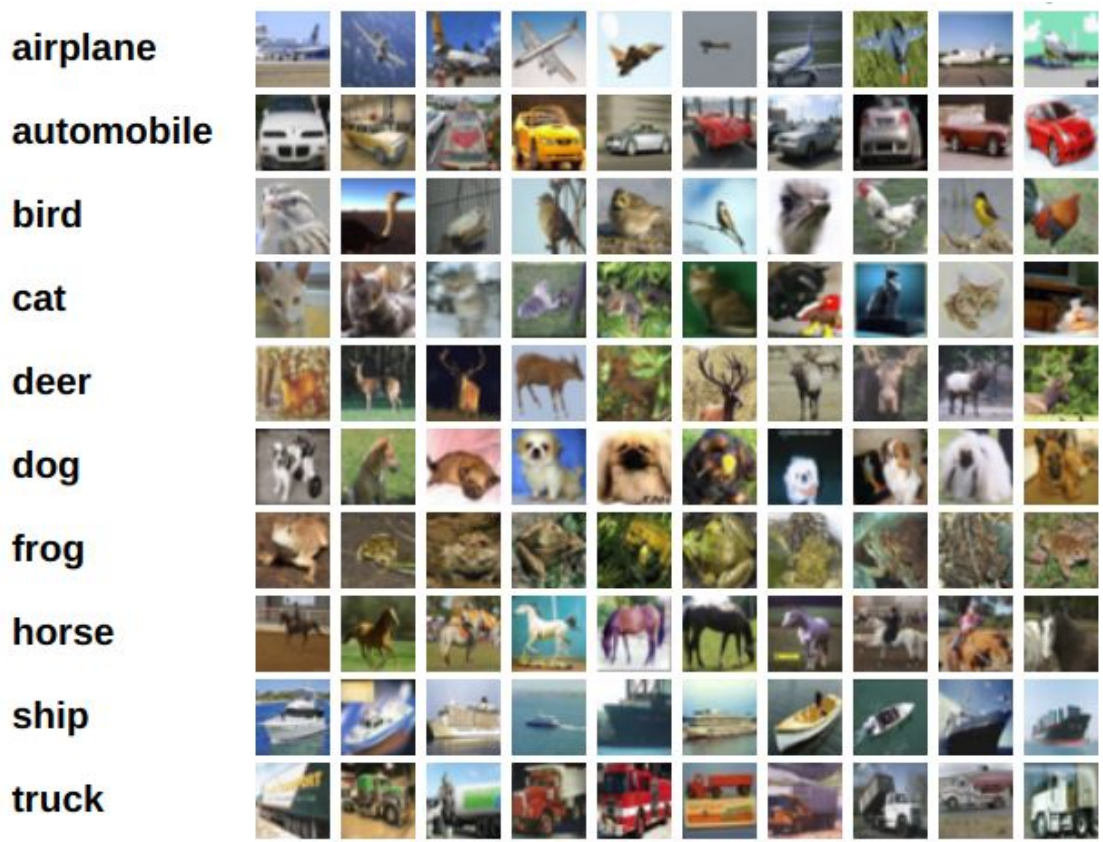
- Like MNIST, but more fashionable (and a little harder)
- 10 categories (sneaker, shirt, boot, etc.)
- Grayscale images
- "If it doesn't work on MNIST, it won't work at all", they said. "Well, if it does work on MNIST, it may still fail on others."



https://github.com/zalandoresearch/fashion-mnist

CIFAR-10

- 60 000 RGB images
- 32x32 (low resolution)
- 10 classes with 6000 images per class



https://www.cs.toronto.edu/~kriz/cifar.html

CIFAR-100

- 60 000 RGB images
- 32x32 (low resolution)
- 100 classes with 600 images

Classes

beaver, dolphin, otter, seal, whale
aquarium fish, flatfish, ray, shark, trout
orchids, poppies, roses, sunflowers, tulips
bottles, bowls, cans, cups, plates
apples, mushrooms, oranges, pears, sweet peppers
clock, computer keyboard, lamp, telephone, television
bed, chair, couch, table, wardrobe
bee, beetle, butterfly, caterpillar, cockroach
bear, leopard, lion, tiger, wolf
bridge, castle, house, road, skyscraper
cloud, forest, mountain, plain, sea
camel, cattle, chimpanzee, elephant, kangaroo
fox, porcupine, possum, raccoon, skunk
crab, lobster, snail, spider, worm
baby, boy, girl, man, woman
crocodile, dinosaur, lizard, snake, turtle
hamster, mouse, rabbit, shrew, squirrel
maple, oak, palm, pine, willow
bicycle, bus, motorcycle, pickup truck, train
lawn-mower, rocket, streetcar, tank, tractor

https://www.cs.toronto.edu/~kriz/cifar.html

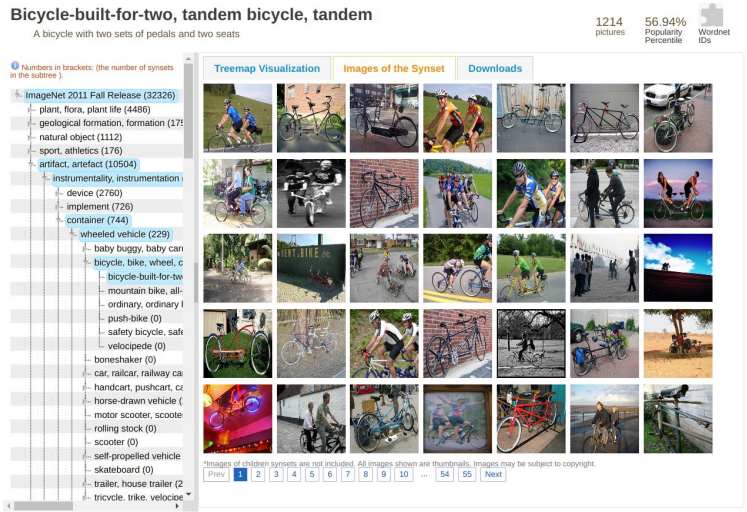
SVHN

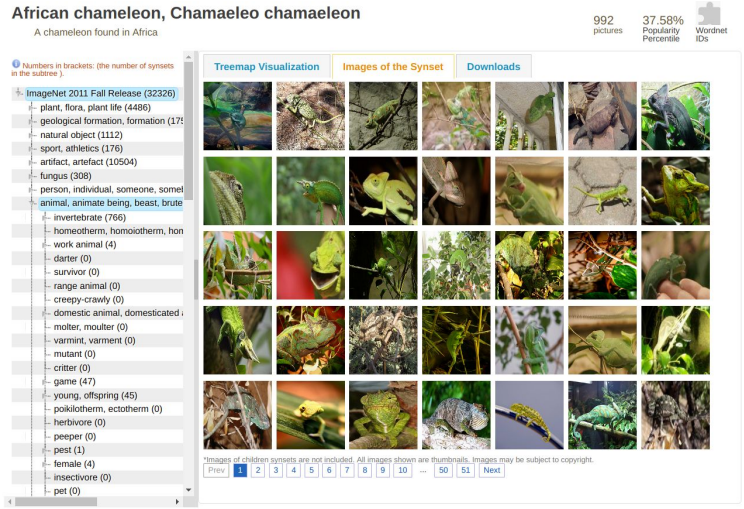
- Street View House Number dataset
- Like MNIST - but harder
- Bounding boxes are provided for each digit
- Approximately 600 000 labelled digits
- Not very high resolution

http://ufldl.stanford.edu/housenumbers/

ImageNet

ImageNet is one of the most widely used datasets in image classification. It has paved the way for many important discoveries in CNN architectures.

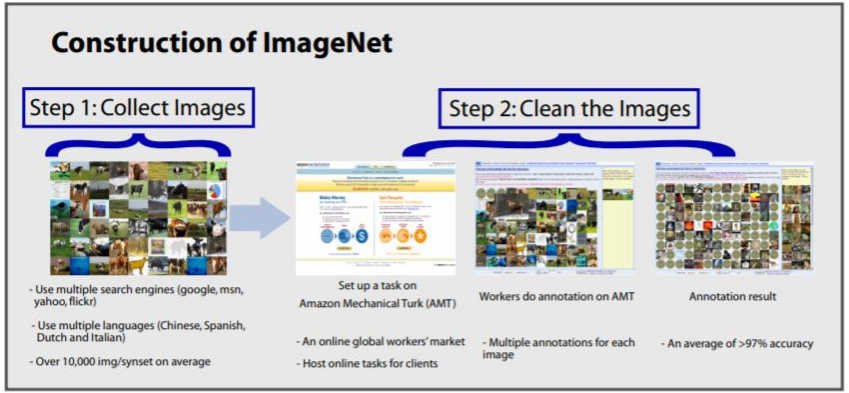




http://imagenet.stanford.edu/

ImageNet

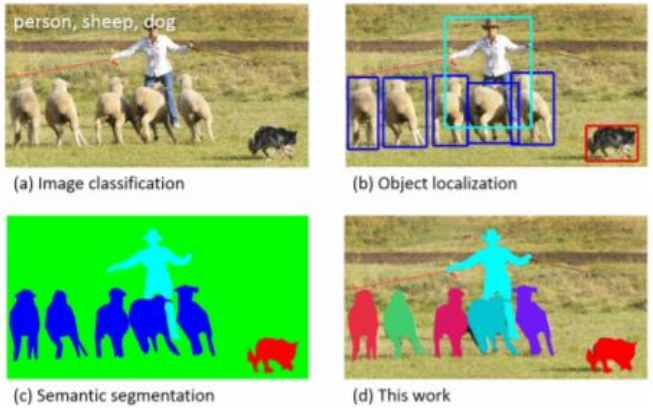
- RGB images with 256x256 resolution
- It consists of 3.2 **million images** manually labelled with thousands of categories.
- ImageNet leveraged crowd-sourcing of data labelling



http://www.image-net.org/papers/ImageNet_VSS2009.pdf
https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/

MS COCO Dataset

- **C**ommon **O**bjects in **CO**ntext
- Instance Segmentation dataset
- **328 000** images with 2.5 million labeled instances and 91 common object categories



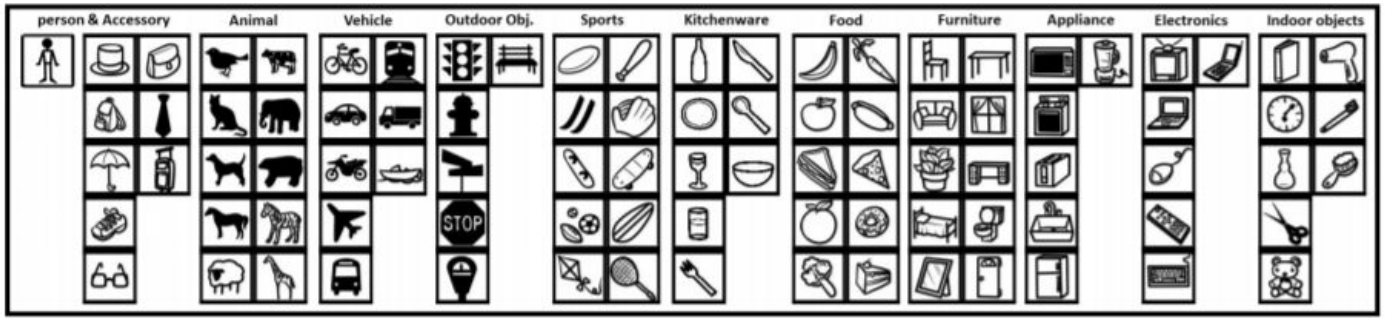
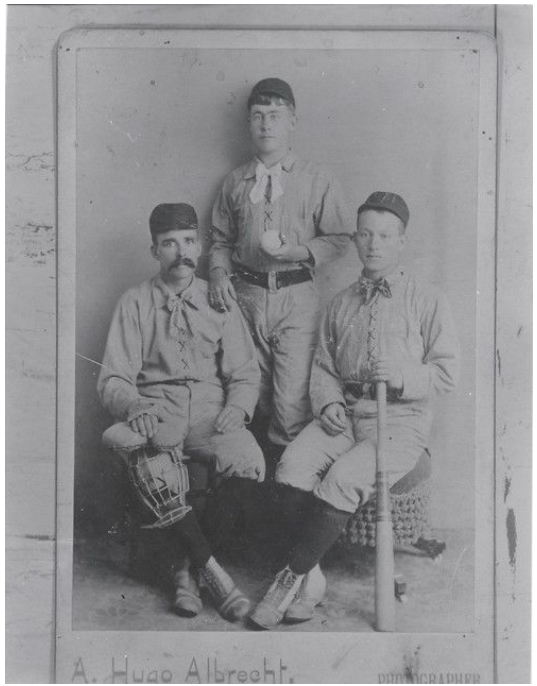


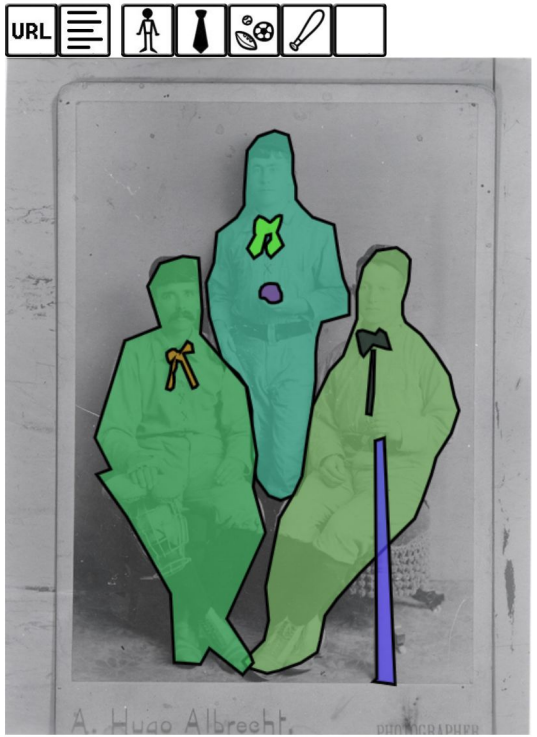
Fig. 11: Icons of 91 categories in the MS COCO dataset grouped by 11 super-categories. We use these icons in our annotation pipeline to help workers quickly reference the indicated object category.

http://cocodataset.org/#home
https://arxiv.org/pdf/1405.0312.pdf

MS COCO Dataset

- Rich level of detail in instance segmentation





http://cocodataset.org/#explore?id=449921
https://farm9.staticflickr.com/8145/7595393302_58cf5044f8_z.jpg

MS COCO Dataset

- **C**ommon **O**bjects in **CO**ntext
- Keypoint detection dataset
- Over 150,000 people and 1.7 million labeled keypoints

http://cocodataset.org/#keypoints-2019

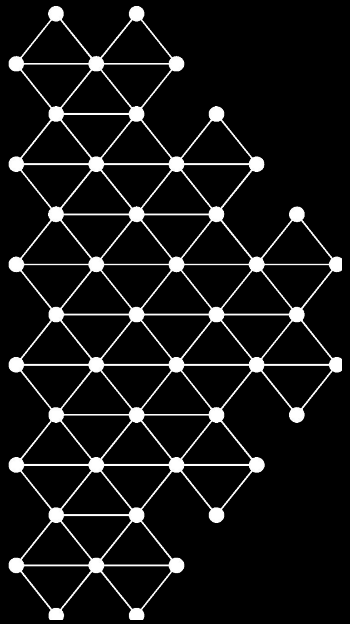
Collecting your own dataset

- Look for similar datasets first
- Understand the kind of labels you need and how you want to represent them
- Labelling can be expensive and inaccurate
- Synthetic data (simulators, GANs, etc.) can work well to test ideas - it can also fail to generalize
- Be sure to have a robust and well guarded test set
- Always challenge the validity of your data

https://medium.com/s/story/no-happy-little-accidents-8663540763f8

Quebec
Artificial
Intelligence
Institute

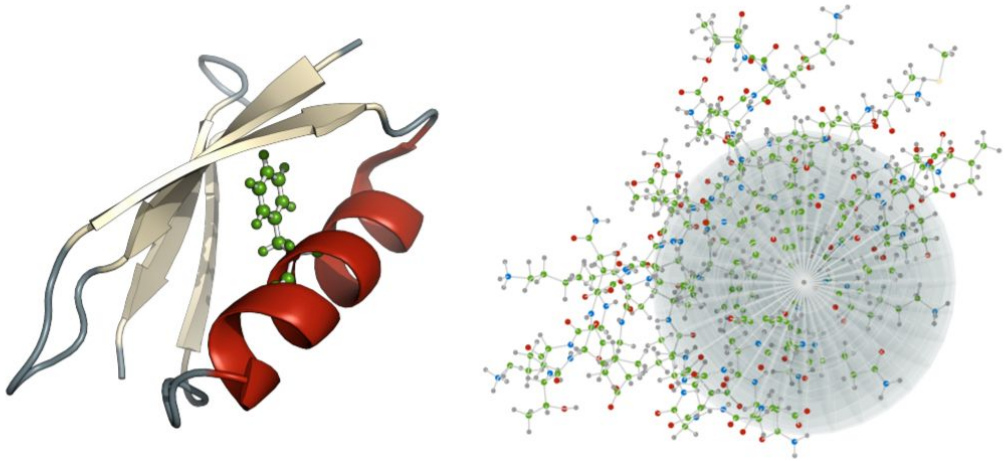
Mila



Questions?

Molecular Properties

Spherical CNNs (an extension of CNNs) have shown lots of promise in predictions of properties in molecular and quantum chemistry.



https://papers.nips.cc/paper/6935-spherical-convolutions-and-their-application-in-molecular-modelling.pdf