The Institute of Cancer Research

Introduction to Machine Learning and Deep Learning

16 May 2018

Igor Kozin, Priya Narayanan, Sebastian Pölsterl, Shan Raza

Plan

- 25/04 Introduction
- 02/05 Calculus, Linear Algebra, Linear Models, Logistic Regression
  NumPy
- 09/05 SVM; k-fold Cross-Validation, Boosting
  Scikit-learn
- **16/05** CNNs; Backprop; Representation Learning; Regularisation; SGD
  Keras
- 23/05 Image classification using Deep Learning models
  Tensorflow and TF-tensorboard

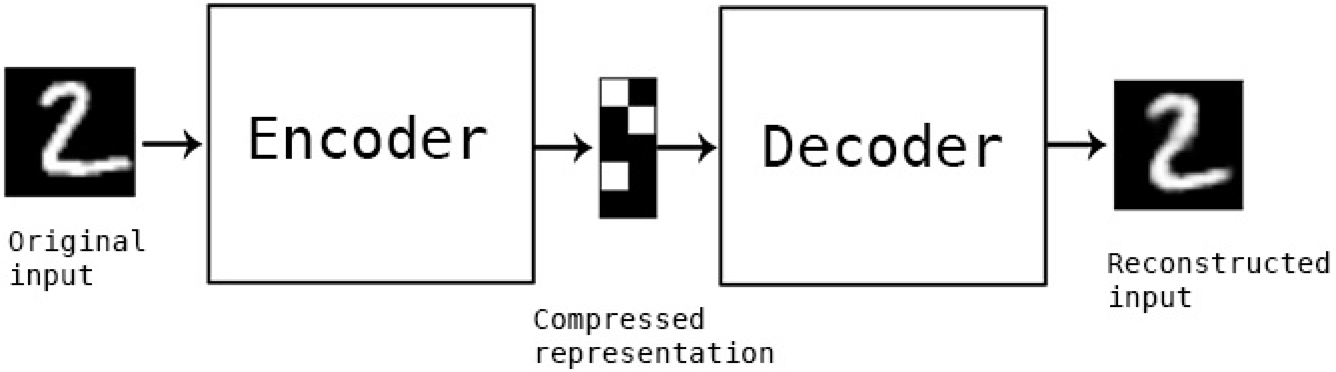
https://github.com/ink1/dl-training

Supervised learning in general

- Input data $X = \{X_1, \dots, X_N\}, X_i \in \mathbb{R}^n$
- Output data $Y = \{Y_1, \dots, Y_N\}, Y_i \in \mathbb{R}^m$
- Find mapping $\mathcal{F}(X_i; \omega) = Y_i{}'$
- such that the loss function $\mathcal{G}(\omega) = \Sigma_i G(Y_i, Y_i{}'; \omega)$
- is minimal over the parameter space $\omega \in \mathbb{R}^k$

Self-Supervised learning
Auto-encoder

- Input data $X = \{X_1, \ldots, X_N\}, X_i \in \mathbb{R}^n$
- Output data $X = \{X_1, \ldots, X_N\}, X_i \in \mathbb{R}^n$
- Find mapping $\mathcal{F}(X_i; \omega) = X_i{}'$
- such that the loss function $\mathcal{G}(\omega) = \Sigma_i G(X_i, X_i{}'; \omega)$
- is minimal over the parameter space $\omega \in \mathbb{R}^k$



Learning process

Install Keras

```
# we assume you have conda installed - see lecture 1
# create a new environment
~$ conda create -n keras python=3.6
...
~$ conda activate keras
...
(keras) ~$ conda install keras
...
(keras) ~$
# if you have access to GPU you can install tensorflow with GPU support prior
# to installing Keras because otherwise you'll get TF with CPU support only
(keras) ~$ conda install tensorflow-gpu keras
...
~$
# pydot graphviz may be needed
```

Keras – a high level library for Deep Learning

- Works atop of Tensorflow, Theano, CNTK
- Allows to easily build a model containing layers, loss function and optimiser
- Supports GPUs and CPUs through the underlying libs

Dense layer

- Matrix dot product
- Linear operation
- Stacking dense layers without non-linearity has the same effect as having a single dense layer

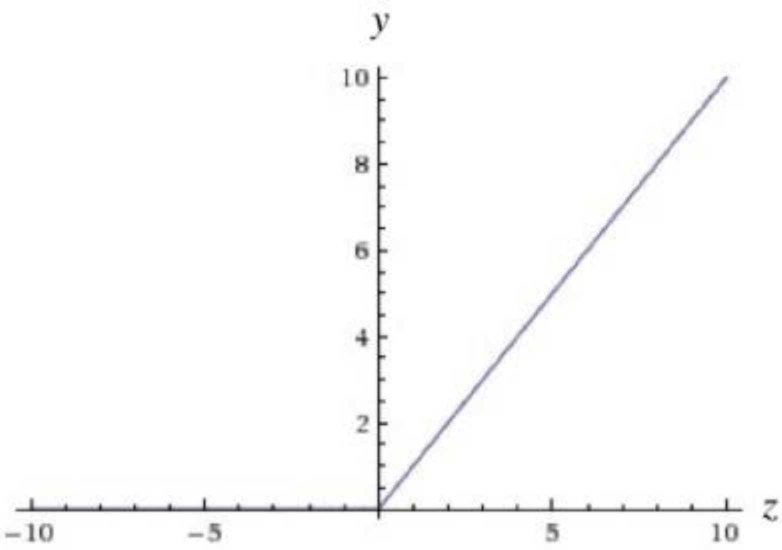
Input: $X = [x_1, \ldots x_N]$

Matrix: $W = [[w_{11}, \ldots w_{1N}], \ldots [w_{M1}, \ldots w_{MN}]]$

Output: $Y = [y_1, \ldots y_M]$ where $y_i = \Sigma_i w_{ij} x_i + b_i$ or $Y = WX + B$

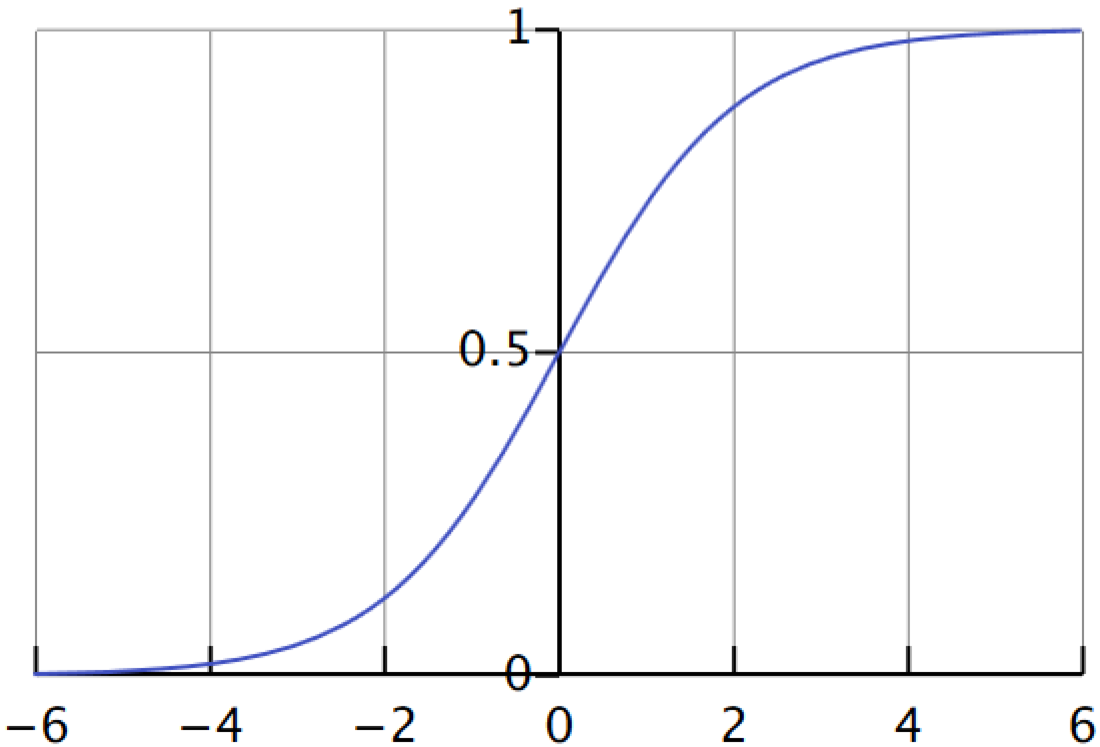
ReLU - Restricted Linear Unit

Nonlinearity is essential in Neural Networks.
ReLU is f(x) = max(0, x)
ReLU is probably the most popular nonlinear unit now.



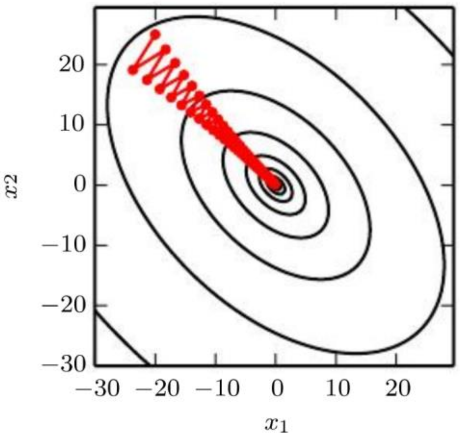
Sigmoid activation

$$S(x) = \frac{1}{1 + e^{-x}}$$

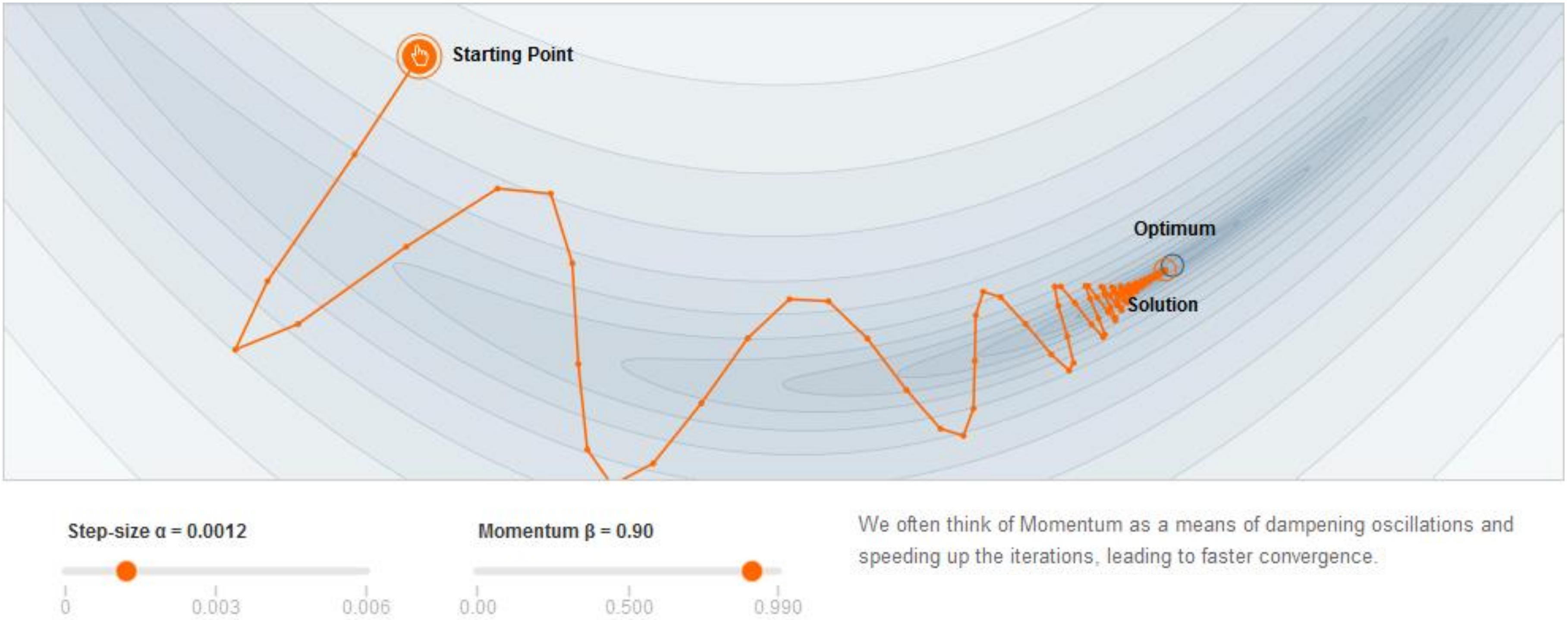


Stochastic Gradient Descent (SGD)

- Assumes the model is differentiable
- Differentiability allows to back-propagate the loss error and to find an improved set of parameters (weights)
- This can be done for all inputs (batch), one input (true SGD) or a randomly selected set of inputs (mini-batch SGD)
- SGD is usually mini-batch SGD
- SGD gives a gradient for improvement but the actual step is determined by learning rate (LR) which is a hyper-parameter
- Batch size is another hyper-parameter
- Small batch sizes help with regularisation but may lead to slower training
- Popular modification is SGD with momentum
- Other variations: RMSprop, Adagrad, Adadelta, Adam see https://keras.io/optimizers/



SGD with momentum



https://distill.pub/2017/momentum/

Basic autoencoder model in Keras

```
autoencoder = models.Sequential()
autoencoder.add(layers.Dense(32, activation='relu', input_shape=(784,)))
autoencoder.add(layers.Dense(784, activation='sigmoid'))
autoencoder.compile(optimizer='sgd', loss='mean_squared_error')
autoencoder.fit(x, x, epochs=50, batch_size=128)
autoencoder.predict(y)
```

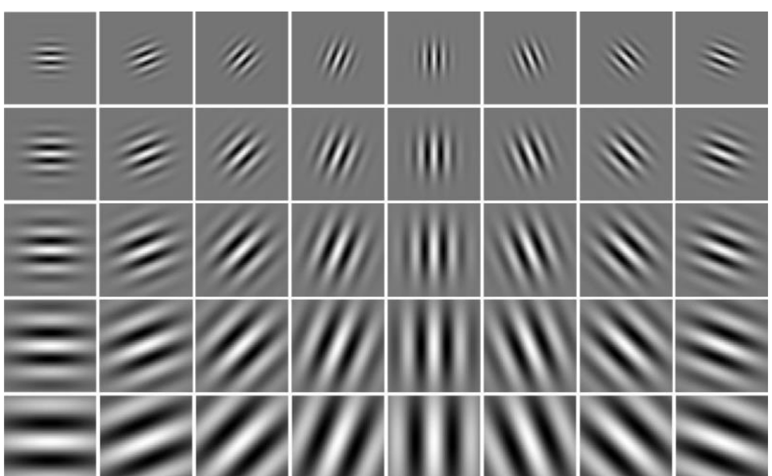
Convolutional Networks

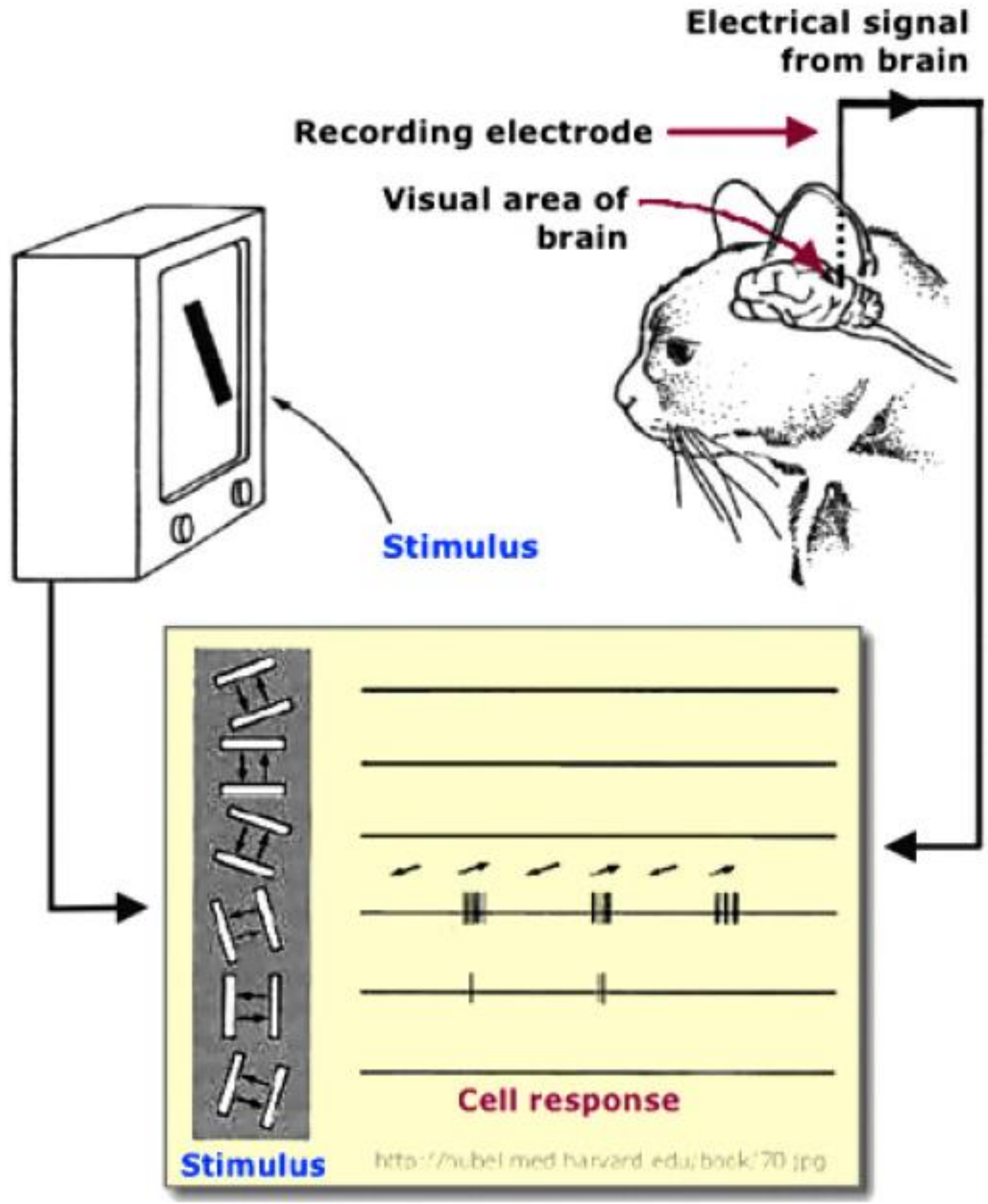
- Features are discovered automatically; layer 1 features resemble Gabor filters; layers represent hierarchy of features

- Training is done iteratively (forward pass is followed by backprop) and may take a long time
- Once the training is done the inference (the forward pass) is usually quick making real time tagging (identification of live stream) possible.

Cat’s visual cortex Hubel and Wiesel

- discovery of neurons responsible for detecting basic image features
- these features can be modelled by Gabor filters in Machine Learning

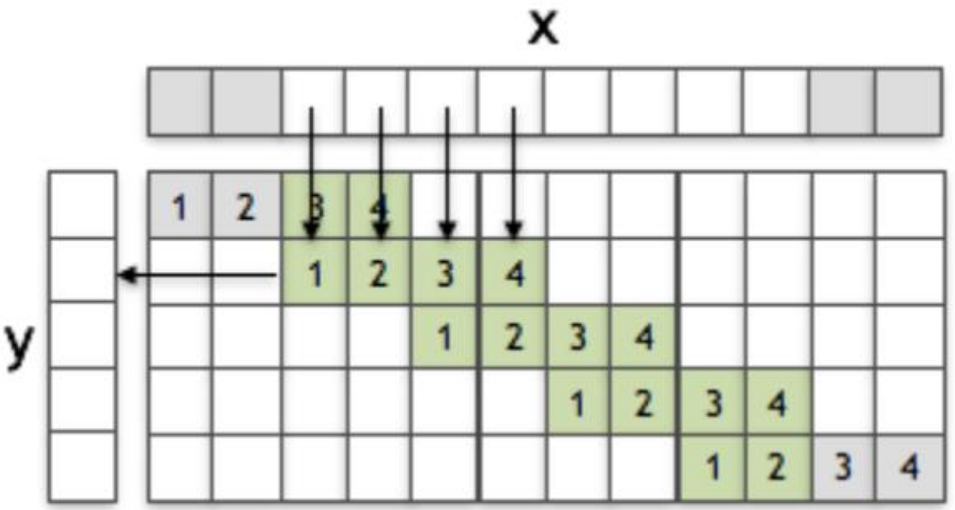




1D convolution

Input vector: x = [1, 2, 1, 3, 2, 0, 2, 1]
Conv kernel: k = [1, 0, 1, -1]

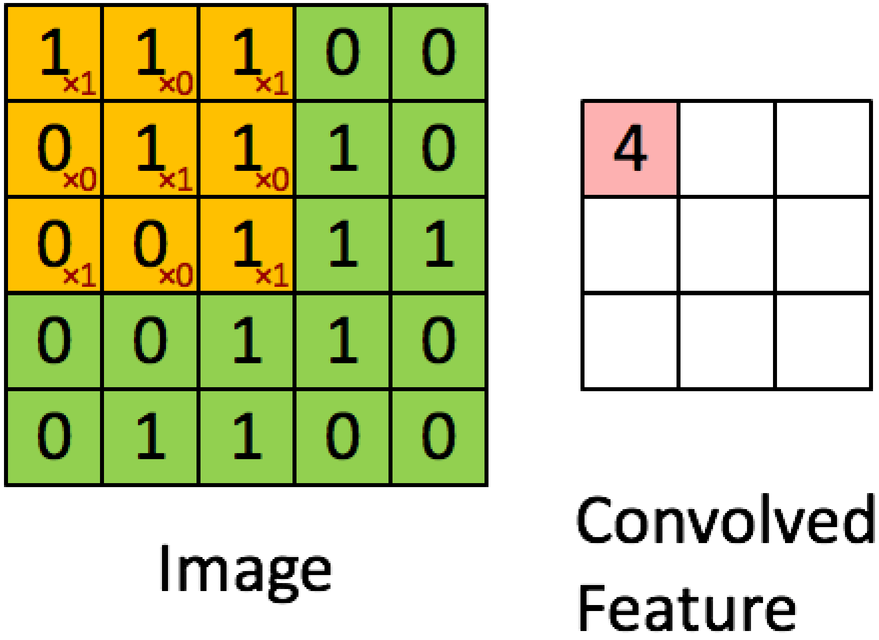
Elementwise product
y_0= [0, 0, 1, 2] x [1, 0, 1, -1] = 0+0+1-2 = -1
y_1= [1, 2, 1, 3] x [1, 0, 1, -1] = 1+0+1-3 = -1
y_2= [1, 3, 2, 0] x [1, 0, 1, -1] = 1+0+2+0 = 3
y_3= [2, 0, 2, 1] x [1, 0, 1, -1] = 2+0+2 -1 = 3
y_4= [2, 1, 0, 0] x [1, 0, 1, -1] = 2+0+0+0 = 2



Output vector: y = [-1, -1, 3, 3, 2]

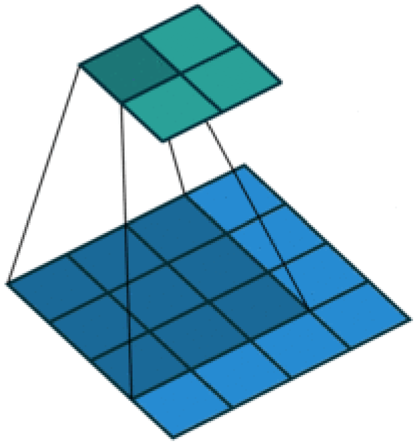
2D convolution

2D convolution kernel:
[[1, 0, 1],
 [0, 1, 0],
 [1, 0, 1]]



2D convolution

No padding, no stride (stride 1)

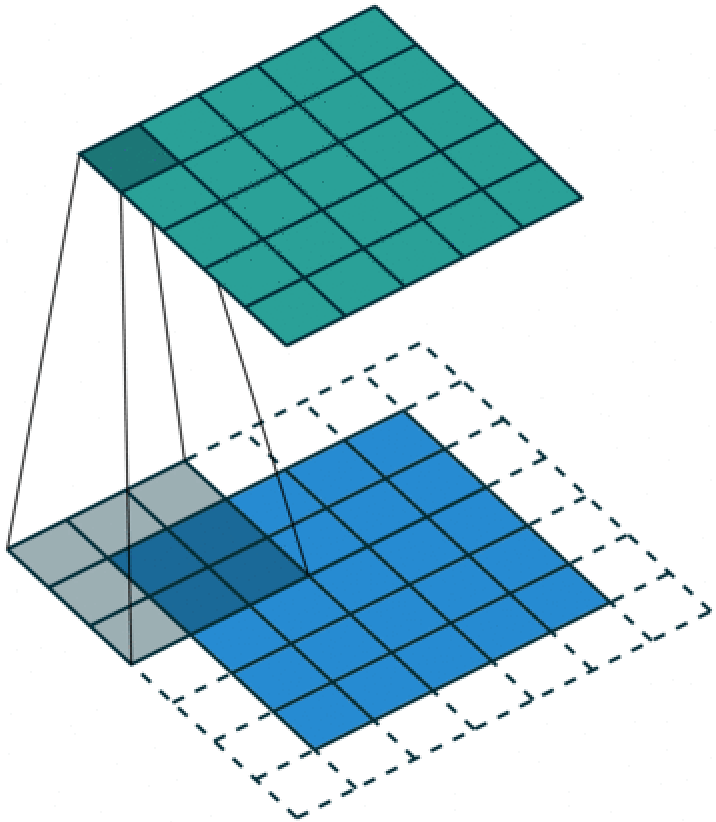


Blue maps are inputs, and cyan maps are outputs.

Credit: https://github.com/vdumoulin/conv_arithmetic

2D convolution

Padding 1, stride 1

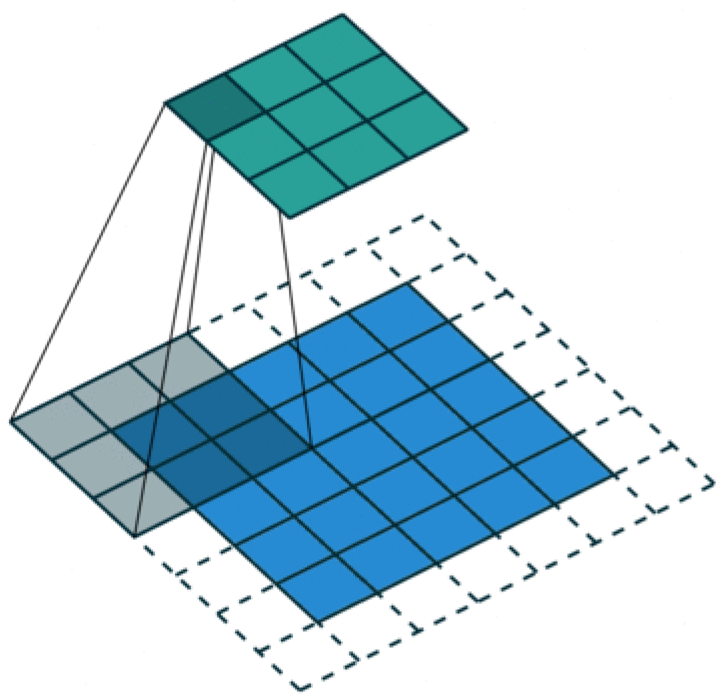


Blue maps are inputs, and cyan maps are outputs.

Credit: https://github.com/vdumoulin/conv_arithmetic

2D convolution

Padding 1, stride 2



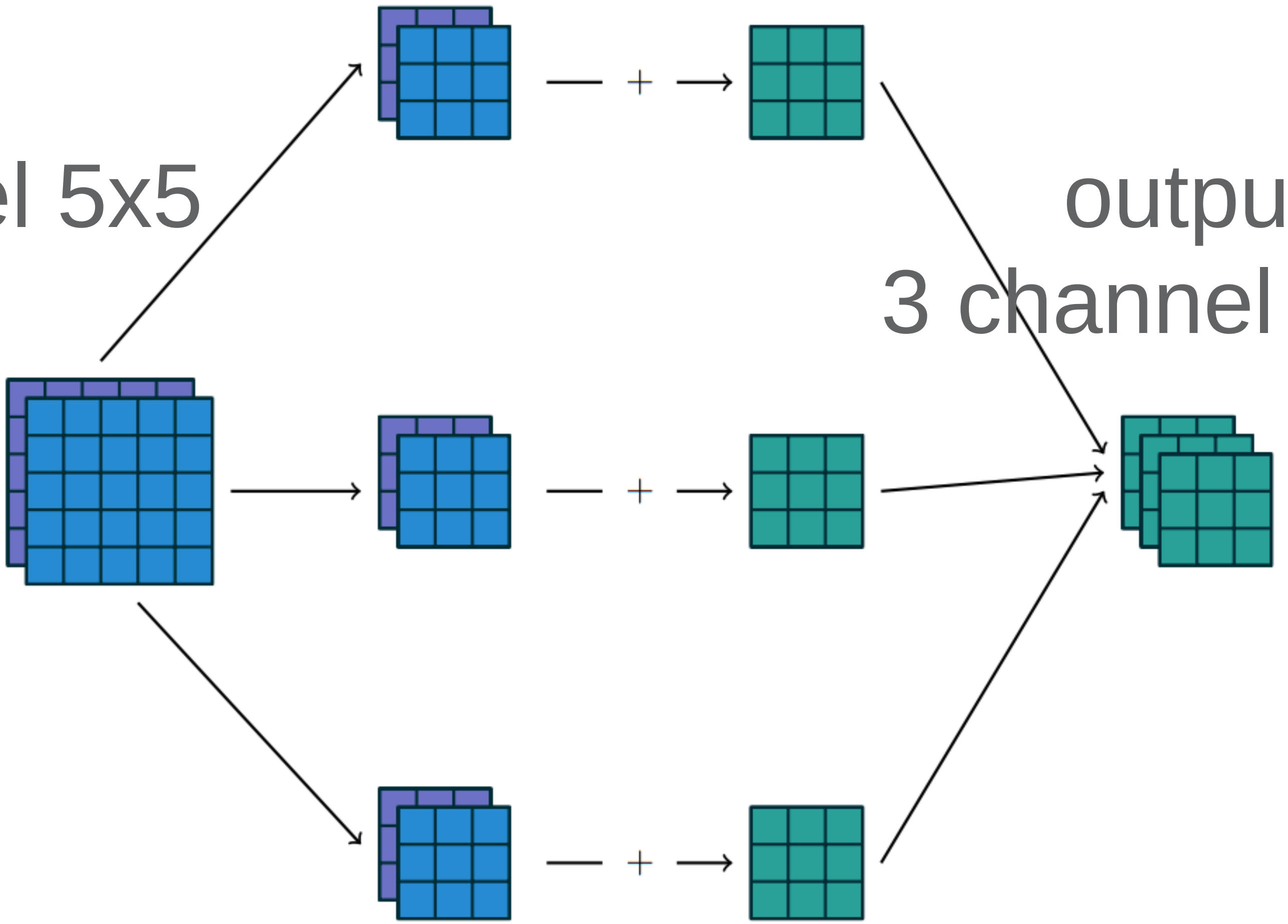
Blue maps are inputs, and cyan maps are outputs.

Credit: https://github.com/vdumoulin/conv_arithmetic

Convolution in action:
3x 2D conv kernels 3x3, no padding, stride 1

input
2 channel 5x5

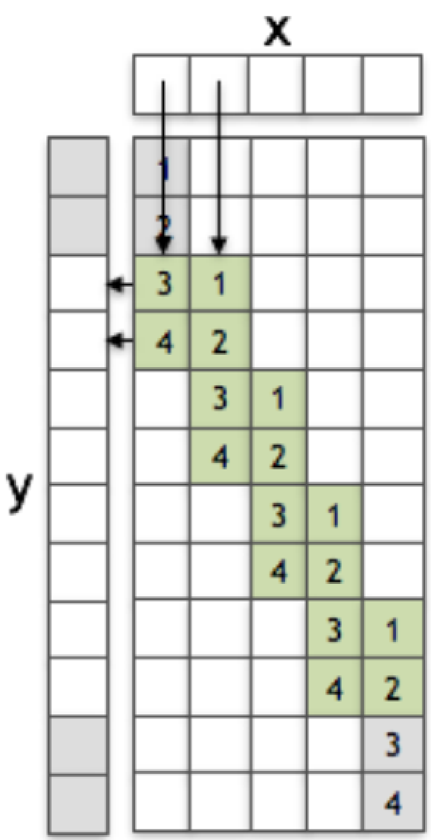
output
3 channel 3x3

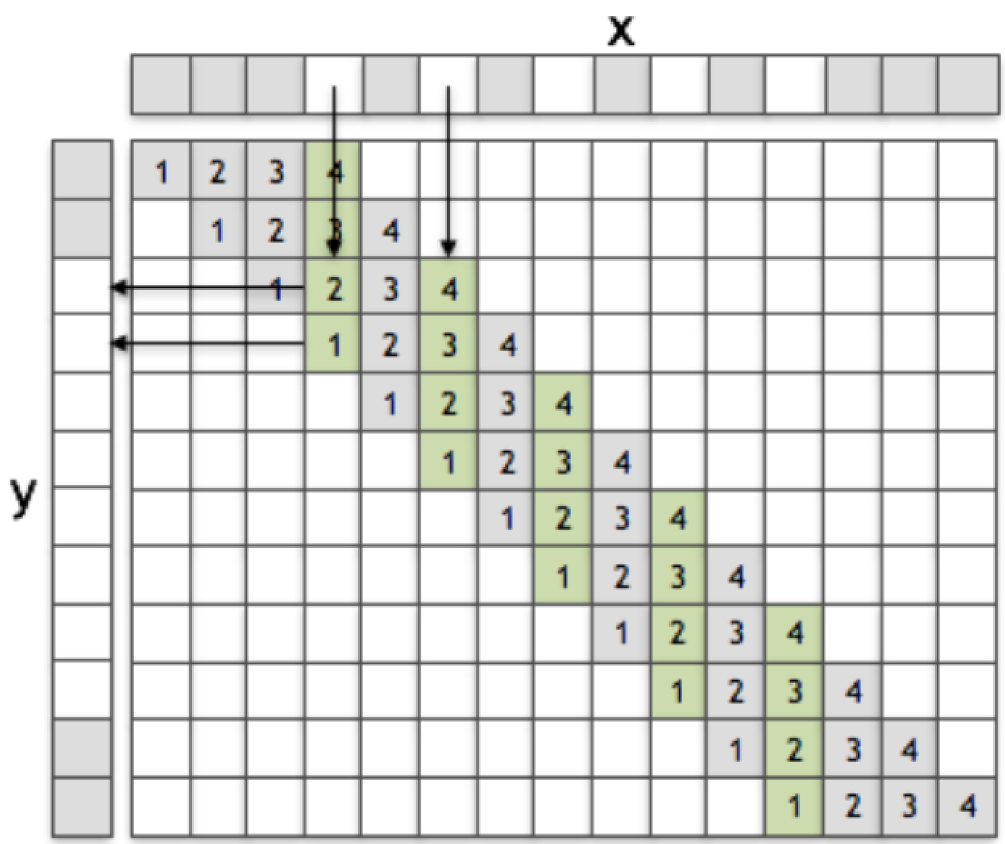


Transposed 1D convolution

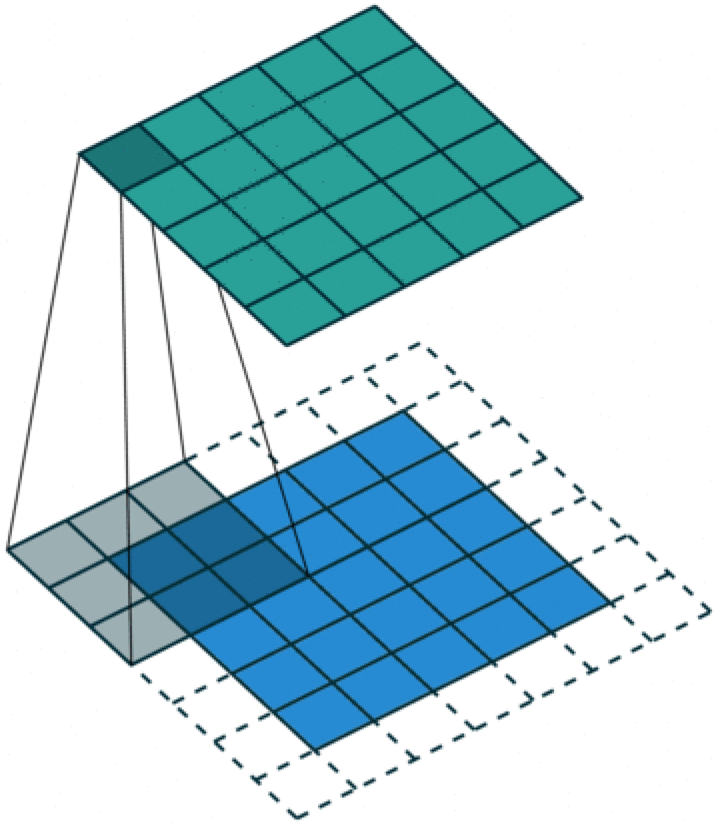
(Deconvolution)

Stride 2 convolution = Sub-pixel convolution with stride 1/2





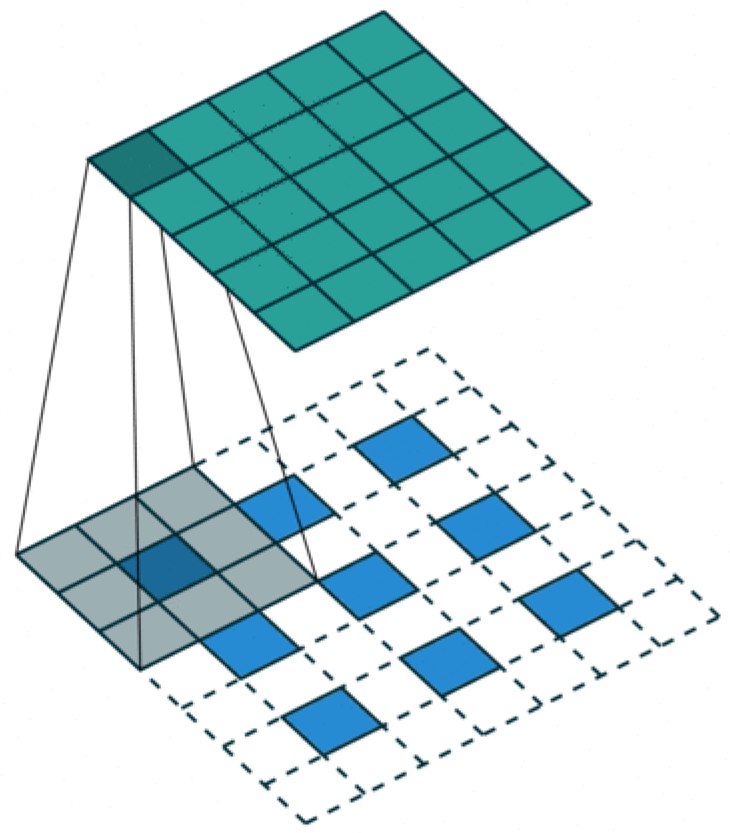
The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 1×1 stride



Blue maps are inputs, and cyan maps are outputs.

Credit: https://github.com/vdumoulin/conv_arithmetic

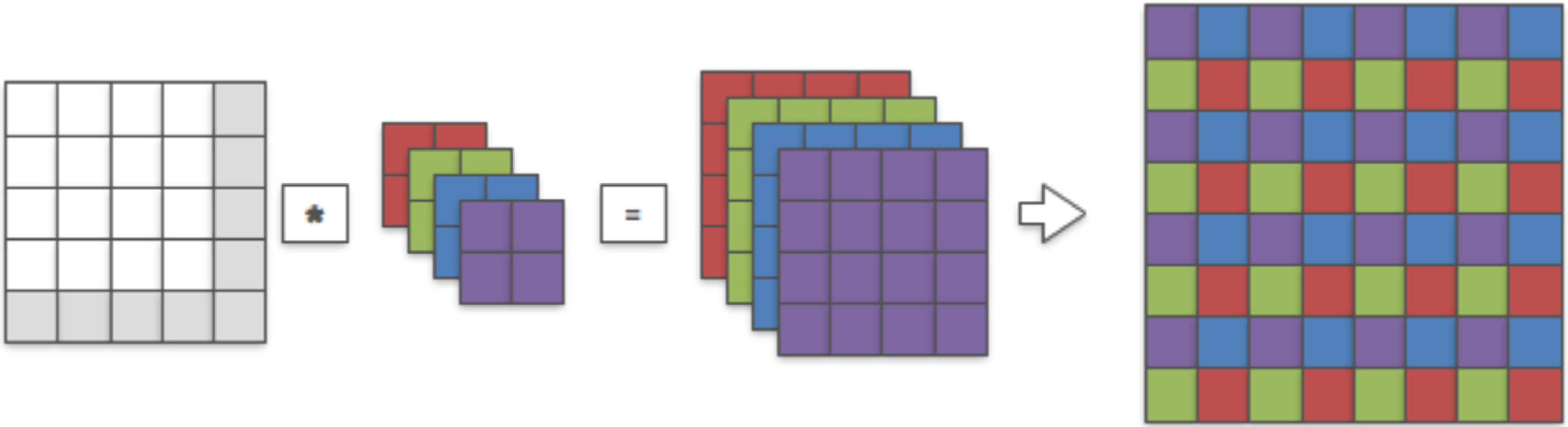
The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 2×2 stride



Blue maps are inputs, and cyan maps are outputs.

Credit: https://github.com/vdumoulin/conv_arithmetic

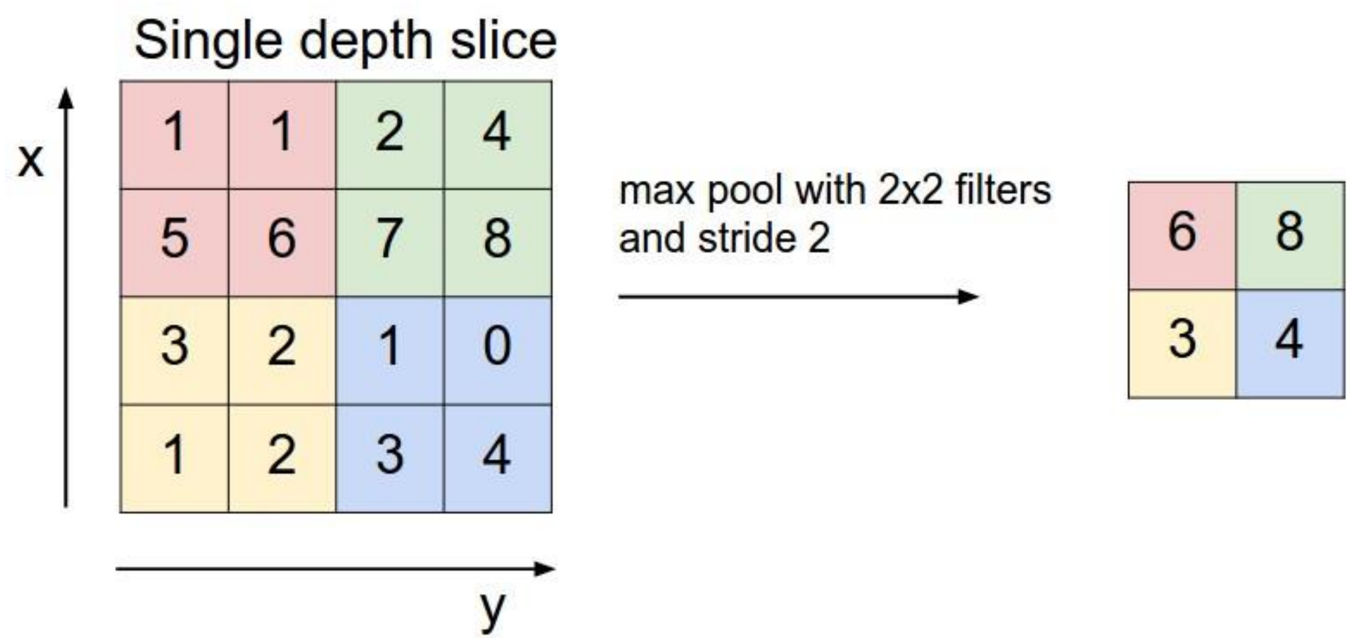
2D subpixel convolution using just convolution



https://arxiv.org/abs/1609.07009

Pooling layer

A sliding window like convolution but with a purpose of shrinking the output size. It usually applies a specific function like maximum or average to the elements in the window.



Regularisation

is a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting (happens when your training loss is much better than validation loss).

- SGD has regularising effect
- drop out
- L2 norm regularisation

$$L' = L + \lambda \Sigma_i w_i^2$$

- L1 norm regularisation

$$L' = L + \lambda \Sigma_i |w_i|$$

Regularisation in Keras

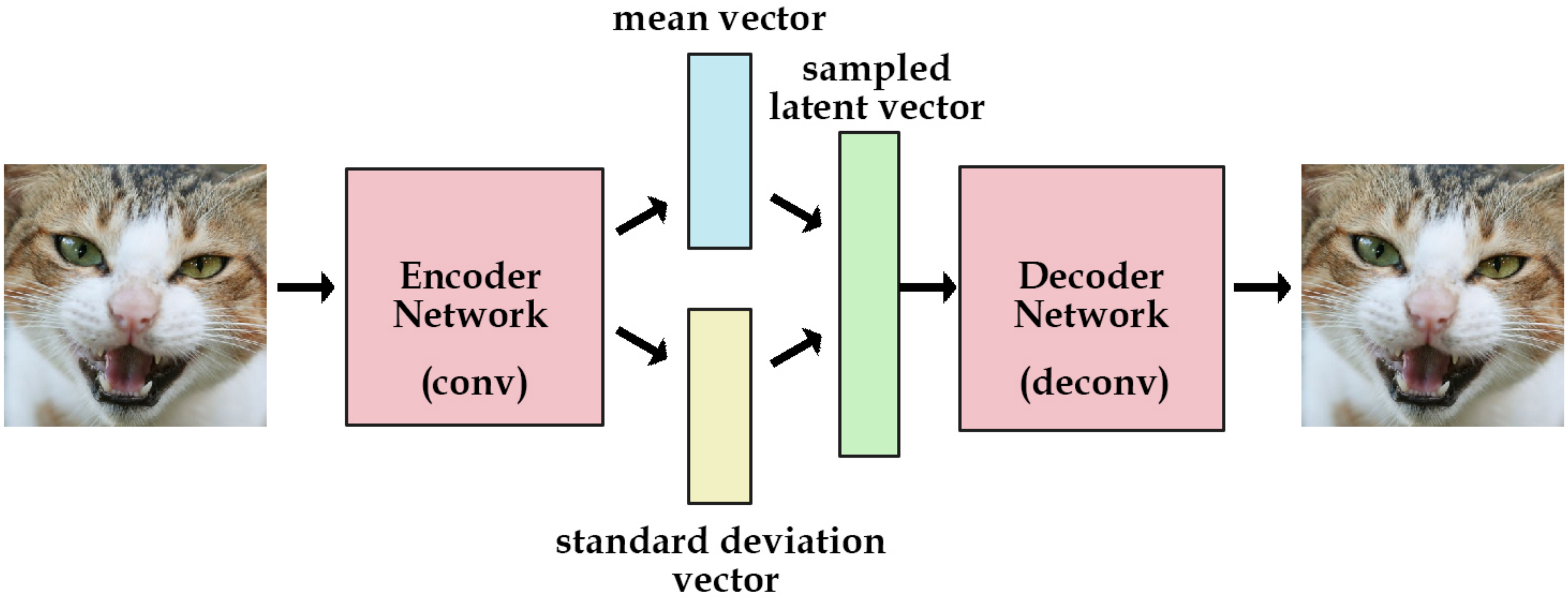
```
# https://keras.io/regularizers/

from keras.layers import Dense, Dropout
from keras import regularizers, models

model = models.Sequential()
model.add(Dense(32, activation='relu', input_shape=(784,),
                kernel_regularizer=regularizers.l2(0.01),
                bias_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(784, activation='sigmoid'))
```

Variational Auto-Encoder (VAE)

generation_loss = mean(square(generated_image - real_image))

latent_loss = KL-Divergence(latent_variable, unit_gaussian)

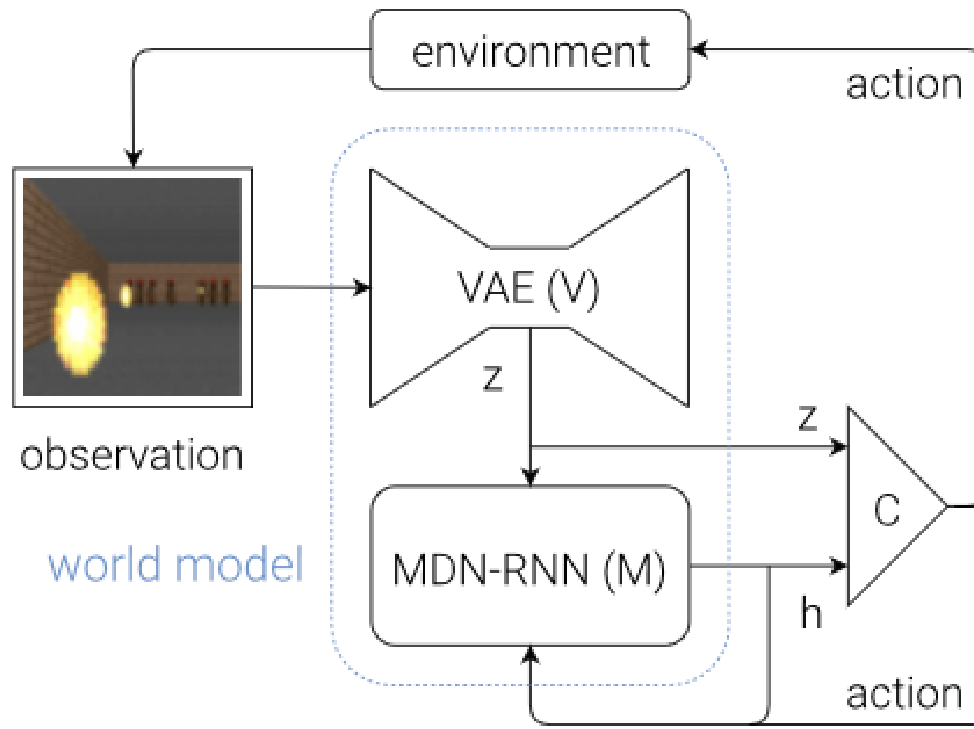
loss = generation_loss + latent_loss



Ha & Schmidhuber, "World Models", 2018

Train an agent to dodge fireballs in Doom (reinforcement learning)

https://worldmodels.github.io/



References and homework

- François Chollet, *Deep Learning with Python*, Manning 2017; Chapter 2: Before we begin: the mathematical building blocks of neural networks and Chapter 3: Getting started with neural networks
https://www.manning.com/books/deep-learning-with-python
- François Chollet, 2016
Building Autoencoders in Keras, tutorial
https://blog.keras.io/building-autoencoders-in-keras.html
- Keras documentation
https://keras.io/layers/convolutional/
- Wenzhe Shi et al, 2016
Is the deconvolution layer the same as a convolutional layer?
https://arxiv.org/abs/1609.07009
- Vincent Dumoulin, Francesco Visin, 2016
A guide to convolution arithmetic for deep learning
https://arxiv.org/abs/1603.07285
- Homework: four notebooks on auto-encoders and optionally one more on variational auto-encoder (VAE)
https://github.com/ink1/dl-training/

ICR