Université de Paris

# Deep learning and applications – Part 3

Sylvain Lobry

13/10/2023

Some slides from d2l.ai

# Before we start…

- https://app.wooclap.com/VMIAI3

# Today’s menu

1. Optimisation
2. Sequence modeling
3. Visual Question Answering
4. Activity

# Before we start…

- You should be familiar with the concepts of:
  - Convolutional layers
  - Pooling layers (max and average)
  - Activation function
  - Loss function
  - Back propagation
  - Gradient descent

## Optimization and tricks

# Stochastic Gradient Descent

Reminder: computation of the gradient (in the supervised learning case):

We have $l$ annotated samples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$ and the empirical risk is defined as:

$$\mathrm{J(f)} = \frac{1}{l}\sum_{i=1}^{l} L(y_i, f(\boldsymbol{x}_i))$$

Where $L(y, \hat{y})$ is the loss function for one sample.

To compute the gradient, we would have:

$$\nabla \mathrm{J(f)} = \frac{1}{l}\sum_{i=1}^{l} \nabla L(y_i, f(\boldsymbol{x}_i))$$

We can see that this is in $\theta(l)$: when the dataset grows, the computation of the gradient grows linearly.

Solution: **sample** $(\mathbf{x}_\mathrm{i}, \mathrm{y}_\mathrm{i}) \in \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$ and do a gradient descent step based on this.

Note: this solution is unbiased (the expectation is the same)

## Optimization and tricks

# Stochastic Gradient Descent

However, in the case of deep learning: billions of parameters to update.

| | **Gradient descent** | **Batch SGD** | **Stochastic gradient descent (SGD)** |
|---|---|---|---|
| Gradient computation | $\theta(l)$ | $\theta(1)$ | $\theta(1)$ |
| Model's updates | $\theta(1)$ | $\theta(l)$ | $\theta(l)$ |

Solution (or compromise): batch gradient descent:

1) sample a batch instead of a single sample
2) compute the gradient on the batch
3) update

Result:

+ Gradient computation is still in constant complexity (= batch size). If hardware can parallelize: same time as for 1 element.

+ Number of updates is greatly reduced (divided by batch size w.r.t. SGD).

+ Variance of the estimate of the gradient is reduced using batches.

Optimization and tricks

# Stochastic Gradient Descent

In practice, you need to select the batch size.

Small batch size: higher number of updates, high variance of the estimate of the gradient

Large batch size: low number of updates, low variance of the estimate of the gradient

Solution: take very large batch??

No, in practice, gradient's computation too long to compute after a certain threshold.

General solution: take a batch as large as your GPU memory can fit.

Takeaway for Batch SGD:

- Faster than GD
- Faster than SGD with less variance on the gradient estimation
- However, there is still some variance…

Optimization and tricks

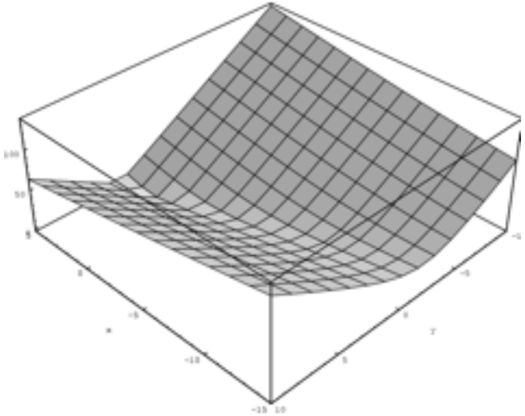
# Momentum

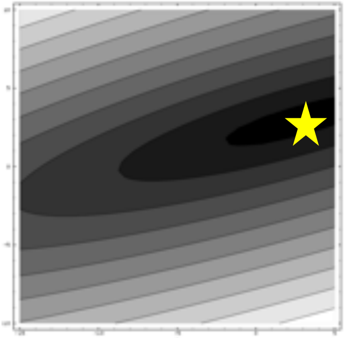
Gradient descent (general rule), with $\epsilon$ the learning rate:

$$w(t+1) = w(t) \; - \epsilon \nabla \mathrm{J}(\mathrm{f}(\mathrm{w}(\mathrm{t})))$$

Two problems:

- Learning rate super important!
- Because we only estimate the gradient (when using Batch SGD), it can be noisy (i.e. variance in the estimation).





Optimization and tricks

# Momentum

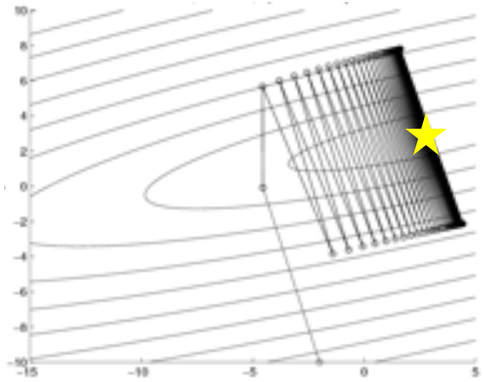
Gradient descent (general rule), with $\epsilon$ the learning rate:

$$w(t+1) = w(t) \ - \epsilon \nabla \mathrm{J}(\mathrm{f}(\mathrm{w}(\mathrm{t})))$$

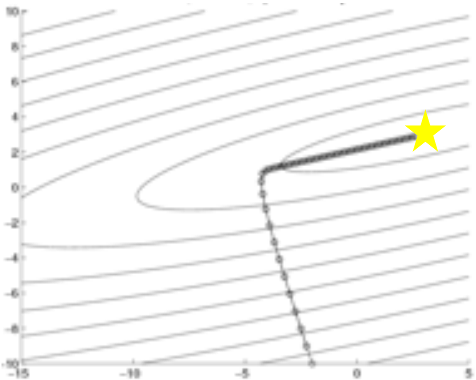
Two problems:

- Learning rate super important!

$\epsilon = 1$: >100 iterations



$\epsilon = 0.1$: 52 iterations



Optimization and tricks

# Momentum

Gradient descent (general rule), with $\epsilon$ the learning rate:

$$w(t+1) = w(t) - \epsilon \nabla J(f(w(t)))$$

Because we only estimate the gradient (when using Batch SGD), it can be noisy (i.e. variance in the estimation).

Solution: leaky average:

$$v(t+1) = \mu v(t) - \epsilon \nabla J\Big(f\big(w(t)\big)\Big)$$
$$w(t+1) = w(t) + v(t+1)$$

I.e. Step of the update is an average of the direction given by the gradient, and the previous direction.

To know more:

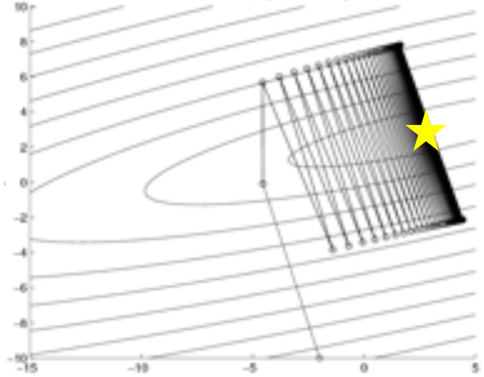
Sutskever I, Martens J, Dahl G, Hinton G. On the importance of initialization and momentum in deep learning. InInternational conference on machine learning 2013 Feb 13 (pp. 1139-1147).

Optimization and tricks

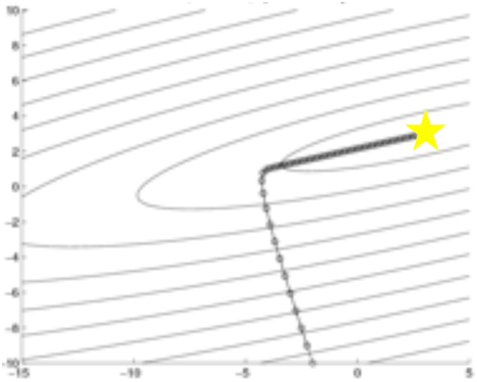
# Momentum

$$v(t+1) = \mu v(t) - \epsilon \nabla \mathrm{J}\Big(\mathrm{f}\big(\mathrm{w}(\mathrm{t})\big)\Big)$$
$$w(t+1) = w(t) + v(t+1)$$

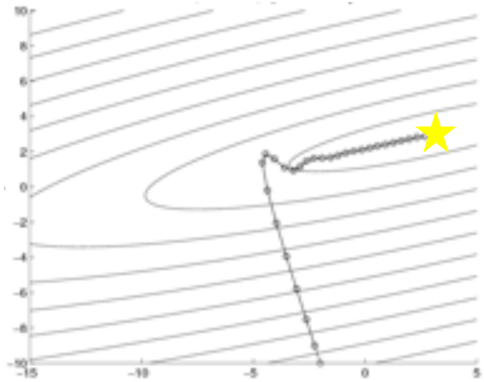
$\epsilon = 1$: >100 iterations



$\epsilon = 0.1$: 52 iterations



$\epsilon = 0.1, \mu = 0.5$: 29 iterations



Optimization and tricks

# Optimization today

- Using SGD with a fixed learning rate: OK for some problems
- Momentum helps in general
- Out of the scope of this class:
  - learning rate scheduling
  - per-coordinate learning rates

- Adam: in general a good choice. Finding good hyperparameters still a trial & error process.

Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

Optimization and tricks

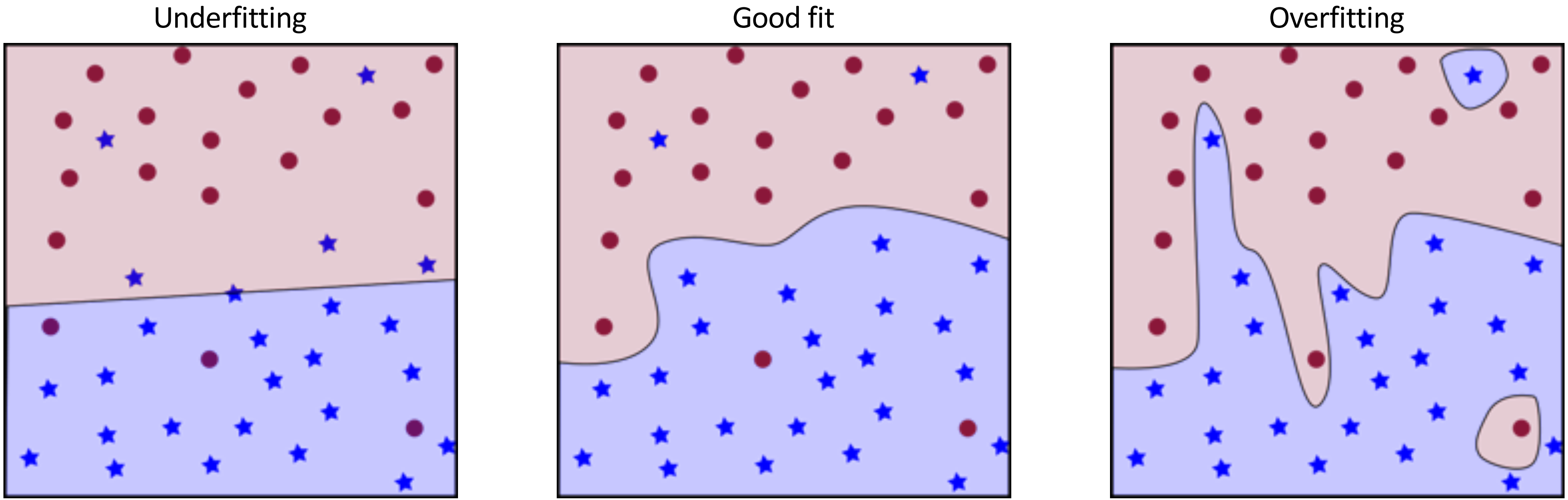
# Questions?

GD vs SGD, Tweet by @ChristophMolnar

Optimization and tricks

# Fighting overfitting

- Neural networks are prone to overfitting



## Optimization and tricks

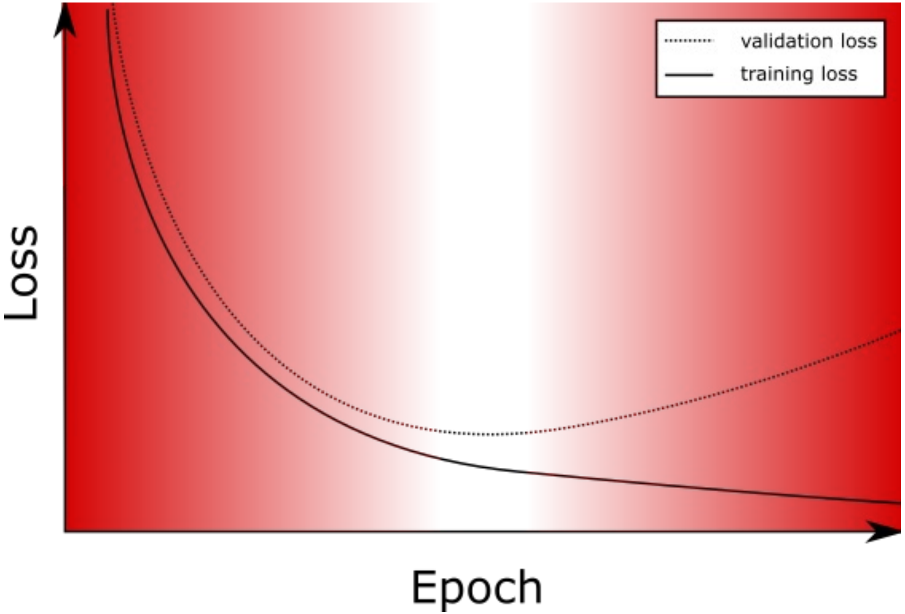
# Fighting overfitting

- Neural networks are prone to overfitting
- One reason: they contain a lot of parameters, can model complex functions
- 1st solution: keep the networks small…
- 2nd solution: have more data…
- Other tricks today

Optimization and tricks

# Are you overfitting?

Underfitting:
- Train longer
- Augment capacity




Overfitting:
- Stop earlier
- Regularize
- Show more data

Optimization and tricks

# Weight decay

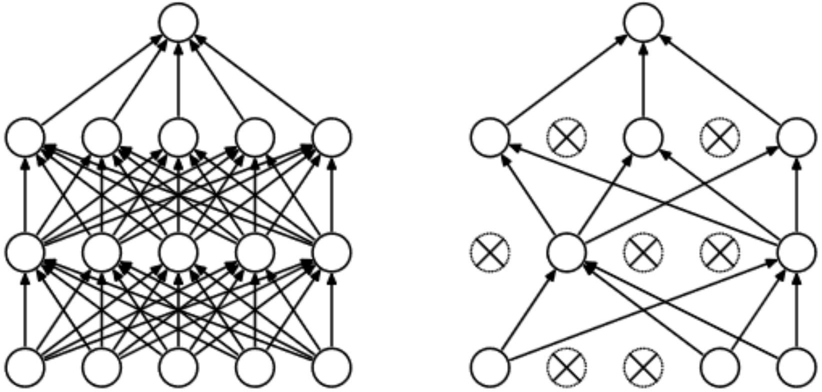
- Hypothesis: you have a big capacity w.r.t. number of training samples
- The model can overfit by having weights that adapts to the training samples. In other words, it can model a function that becomes too complex.
- Solution: penalize too complex models!

- Weight decay:
  - Measure the complexity of a model: L2 norm of the weights.
  - Penalize the complexity: add the L2 norm of the weights to the loss
  - Result: weights shrink towards 0

Optimization and tricks

# Dropout

- One possible reason from overfitting: co-adaptation
- Several intuitions proposed by authors: role of sex in evolution, spread of conspiracy theories…
- Idea: break co-adaptation by disapling nodes at training time
- For MLP



(a) Standard Neural Net (b) After applying dropout.

Image from Srivastava et. al.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.

## Optimization and tricks

# Dropout

- In practice remove a connection with a given probability
- no preferential path can be learned
- slightly different models are learned at every path
- the final model is a kind of average (= ensemble model)

## Optimization and tricks

# Batchnorm

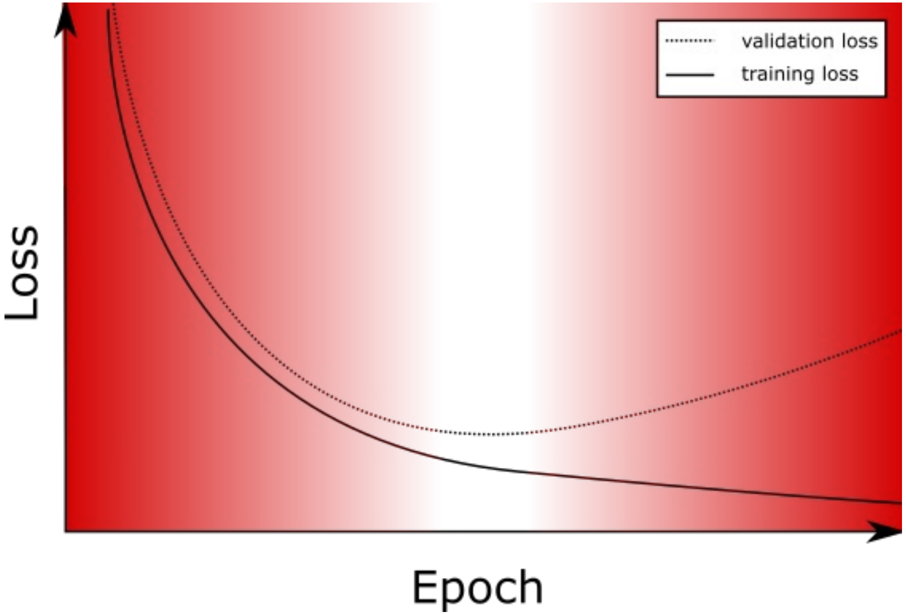
- Intuition: normalize all your feature maps with the statistics you observe within each batch (in general, before the activation function)
- Normalization is always a good thing
- Having a similar scale for all of the inputs allows the network not to have to deal with that
- It will prevent to have a layer dominating because it has high values -> regularization.

- In practice:
  - At training time: normalize w.r.t. the statistics observed over each batch
  - At test time: normalize w.r.t. the statistics computed over the training set.

Optimization and tricks

# Are you overfitting?

Underfitting:
- Train longer
- Augment capacity




Overfitting:
- Stop earlier
- Regularize
- Show more data

Optimization and tricks

# Data augmentation

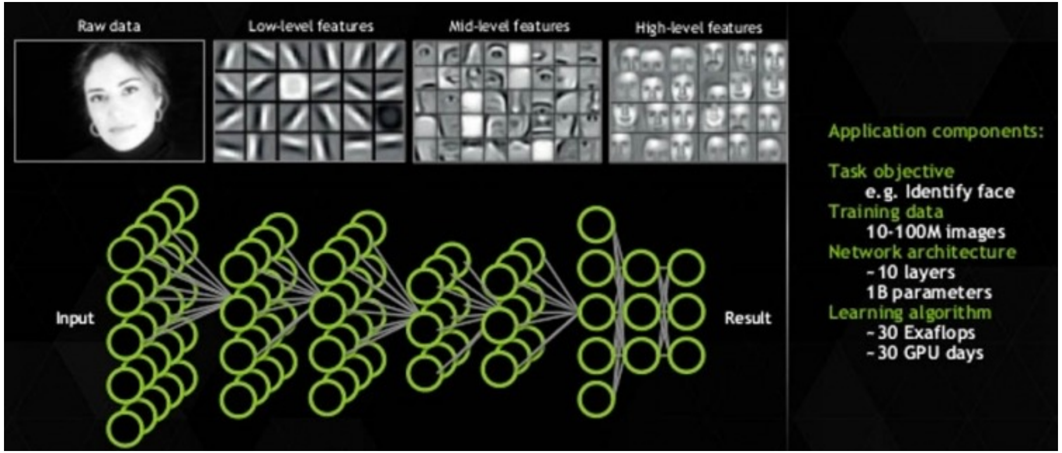
- Set of operations applied randomly to the input data -> obtain different training samples
- Common data augmentation operations for images:
  - Cropping
  - Rotation
  - Vertical/horizontal flips
  - Injecting noise?

## Optimization and tricks

# Fine tuning

- Common technique to train a model when not enough data
- Intuition: a lot of the filters learnt by a model are applicable in most situations (e.g. edge detectors, texture,…)

Credit: NVIDIA



## Optimization and tricks

# Fine tuning

- Common technique to train a model when not enough data
- Intuition: a lot of the filters learnt by a model are applicable in most situations (e.g. edge detectors, texture,…)
- Idea: take a model learnt on a big dataset (e.g. ImageNet)
- Replace the layers you need to change (generally last ones), and initialize them
- Re-train (or fine-tune) everything on your data.

## Today’s menu

1 Optimisation

2 Sequence modeling

3 Visual Question Answering

4 Activity

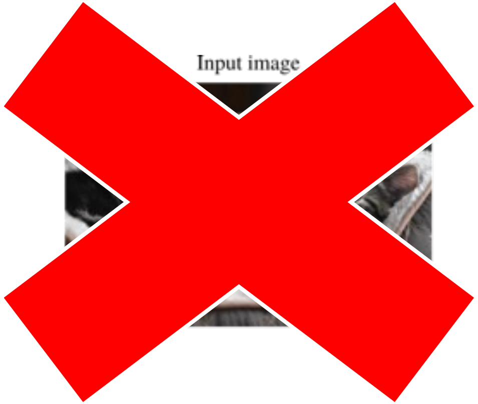
Sequences

# What can you do with an image?

Input image

## Sequences

# What can you do with ~~an image~~ a sequence?



Sequences

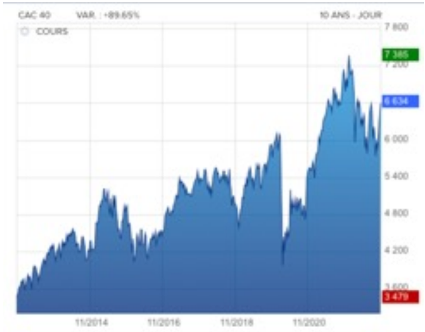
# What can you do with a sequence?

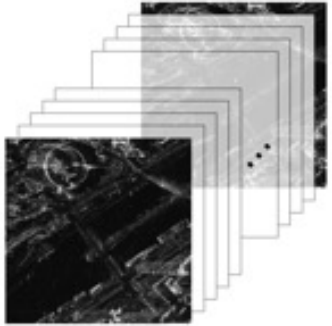
- Up until now: single object $\mathbf{x}$ as input
- Sequence: collection of $T$ objects: $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T$
- Tasks:
  - From a sequence, predict a single value $y$ (e.g. crop type, deforestation, sentiment, …)
  - From a sequence, predict a sequence $y_1, y_2, \dots, y_{T'}$:
    - With 1-to-1 correspondence between $\boldsymbol{x}_1$ and $y_1$ (e.g. presence/absence of forest)
    - Without 1-to-1 correspondence (e.g. machine translation)

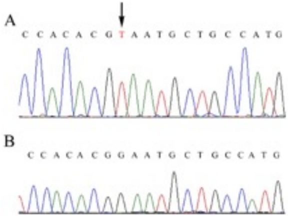
## Sequences

# What can you do with a sequence?

- Examples of sequences:
  - Text
  - Stock market
  - Sequence of images
  - DNA...





Sequences

# Model a sequence

- Model the sequence = model $P(x_1, x_2, \ldots, x_T)$
- We have:

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1}).$$

- If we add a Markovian property:
  - order 0: $P(x_1, x_2, x_3) = P(x_1)P(x_2)P(x_3)$
  - order 1: $P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_2)$
  - order 2: $P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_2, x_1)$
  - …

Sequences

# Model a sequence

- order 1: $P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_2)$
- bigram can be estimated from a corpus:
  - $P(x_2|x_1) = \frac{n(x_2,x_1)}{n(x_1)}$
- Trade-off between modeling long-term dependencies and frequency of co-occurences

Sequences

# Model a sequence

- Ideally, we want to keep the whole sequence in the modeling of each time step, i.e.

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1}).$$

- Computationally impossible to model: too many parameters…

Sequences

# Recurrent Neural Network

- Main idea of RNN:

$$P(x_t \mid x_{t-1}, \ldots, x_1) \approx P(x_t \mid h_{t-1}),$$

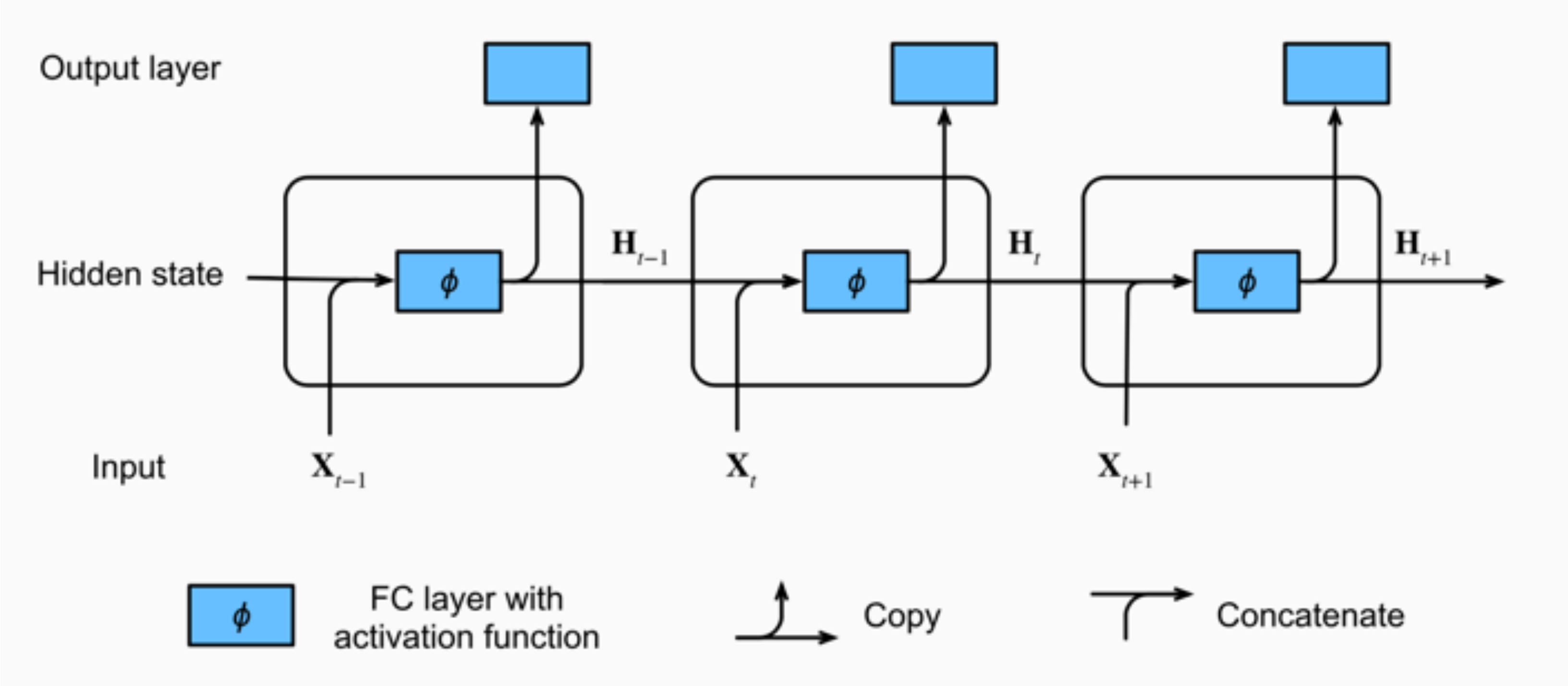
- with:

$$h_t = f(x_t, h_{t-1}).$$

- The variable $h_{t-1}$ is a hidden state: it stores information on the sequence until step t - 1

Sequences

# Recurrent Neural Network




d2l.ai

Sequences

# Dealing with text

- A RNN expect a vector of numbers as an input.
- A text is a string
- Tokenization:
  - “Hello everybody, enjoy the class and enjoy life” ->
  - [“Hello”, “everybody”, “enjoy”, “the”, “class”, “and”, “enjoy”, “life”]
- Words converted to numbers using a vocabulary:
  - Either created from the text: [0, 1, 2, 3, 4, 5, 2, 6]
  - pre-defined (from a large corpus, to use pre-trained models)
- Reversible operation

Sequences

# The problem with this RNN

- Hard to learn long-term dependencies because of exploding and vanishing gradient
- Desirable properties:
  - Important information should be retained after a while (long-term memory)
  - All the information from the recent past should be stored (short-term memory)

- Intuition: the hidden state cannot be computed as a uniform average of all the previous steps

Sequences

# Gates

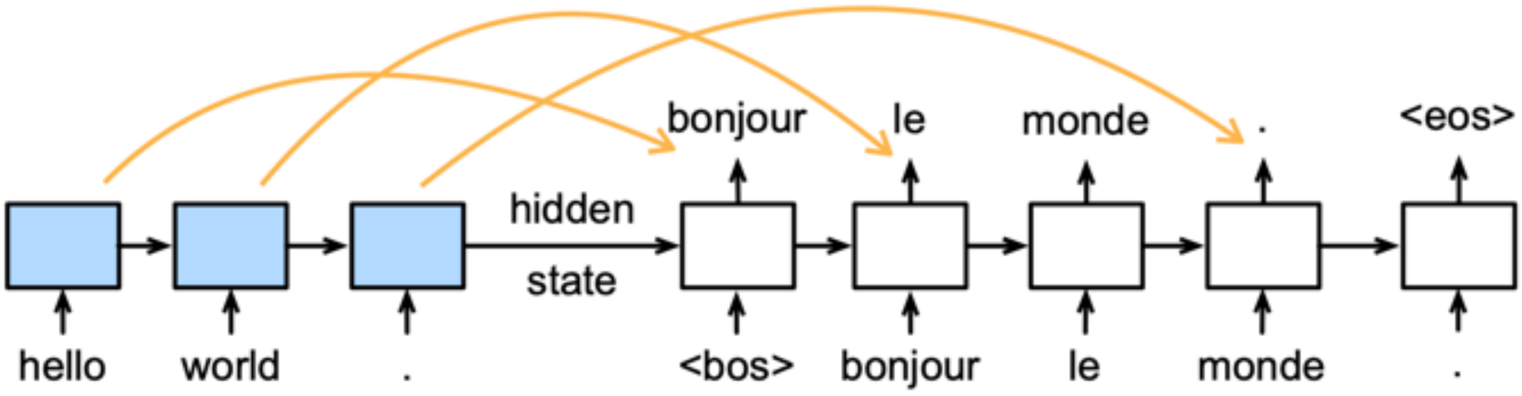
- LSTM (Long-short-term memory) introduces the concept of gate:
- A gate = learnable parameter deciding how much information we want to keep
- Regrouped in a memory cell in charge of computing the gates from an internal state

- In LSTM, 3 gates:
  - input gate: how much the input ($x_t$) should influence the current step?
  - forget gate: how much of the internal state should we keep?
  - output gate: how much the current step should influence the output?

Sequences

# Attention

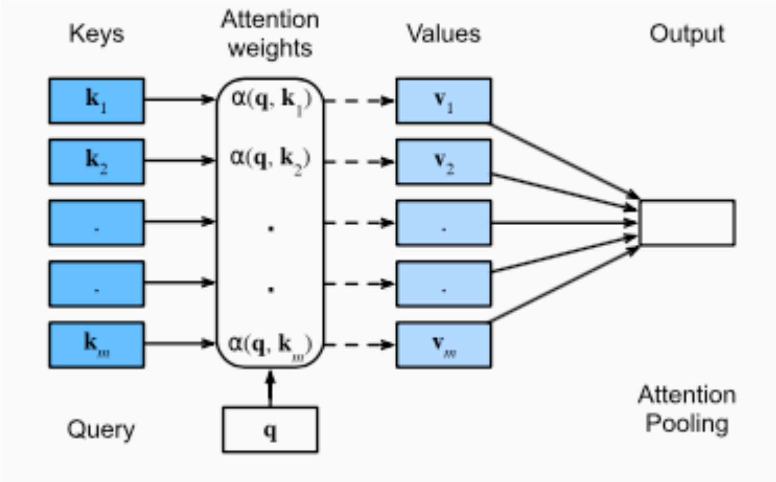
- Let’s take an example of sequence to sequence translation.
- Generated tokens are related to a combination of input tokens



## Sequences

# Attention

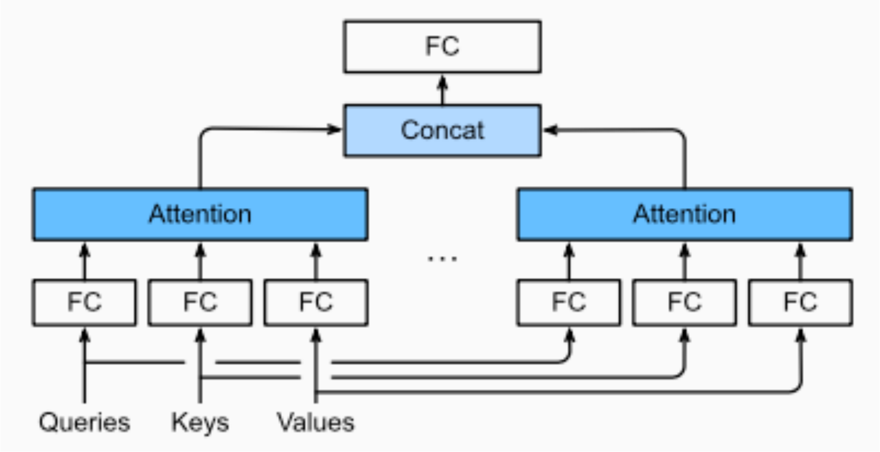
- Attention = make a linear combination of values, based on the combability between a set of keys and a query
- Allows to select relevant input tokens to generate output tokens



Sequences

# Attention

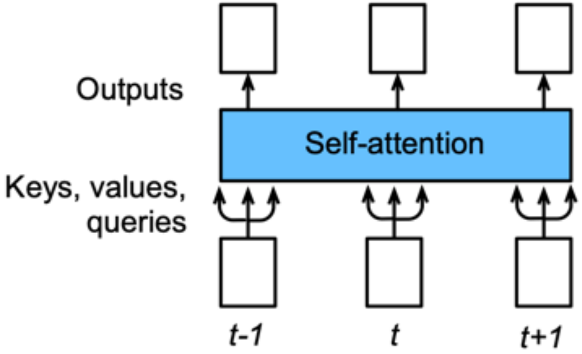
- In practice, it can be useful to capture several dependencies between tokens (e.g. long vs short range).
- This can be done by allowing different representations of queries, keys and values



Sequences

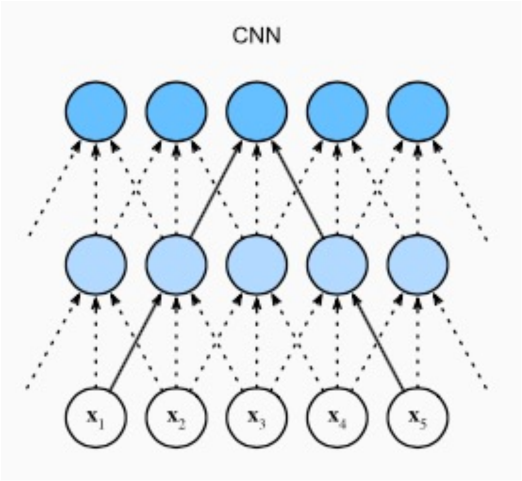
# Self-attention

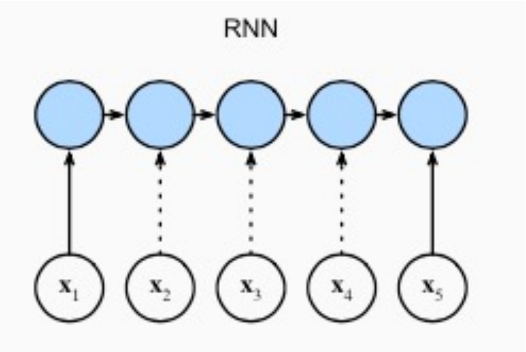
- Let’s feed a sequence of tokens to a model
- Let’s give each token is own set of query, keys and values -> each token can attend (through its query) any combination of others token (through their keys).

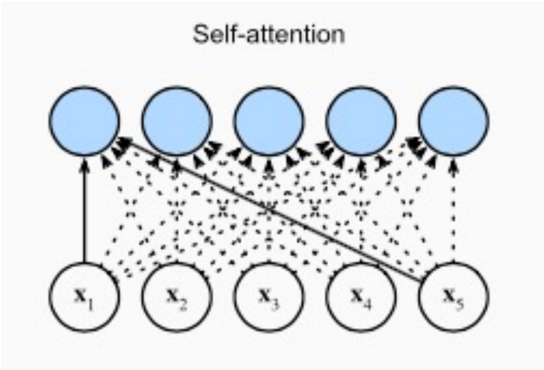


Sequences

# Self-attention



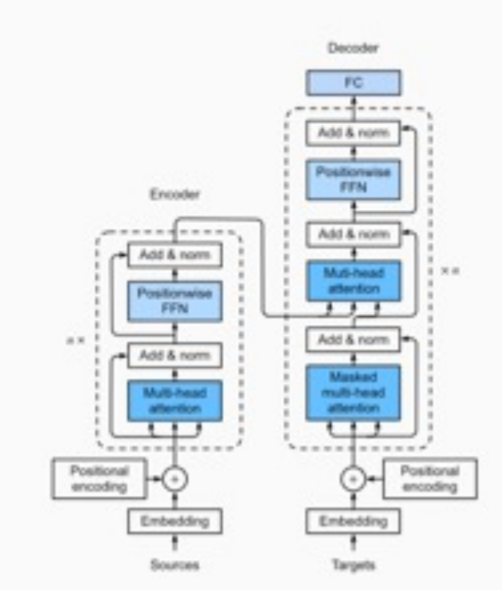



Sequences

# Transformer

- Self-attention + positional encoding -> Transformer architecture




Vaswani, Ashish, et al. "Attention is all you need."
*Advances in neural information processing systems* 30 (2017).

Sequences

# Conclusion

- Very basics of sequence modeling
- RNN still used today, outperformed by transformers
- Transformers allow for interactions of each token with any other -> computationally heavy, but much better modeling of interactions
- Transformers can also be used for vision

## Today’s menu

1 Optimisation
2 Sequence modeling
3 Visual Question Answering
4 Activity

# Visual Question Answering

- Objective: provide an answer (in natural language) from:
  - One image
  - One question (in natural language)

# Samples from the VQA2 dataset (from Teney, 2017)

**What is on the coffee table ?**
*candles*

**What color is the hydrant ?**
*black and yellow*

**What is on the bed ?**
*books*

**What is the long stick for ?**
*whipping*

# Visual Question Answering

- This is a new task! Introduced by [1] in 2015
- Can be seen as an extension of the Turing test to the visual domain

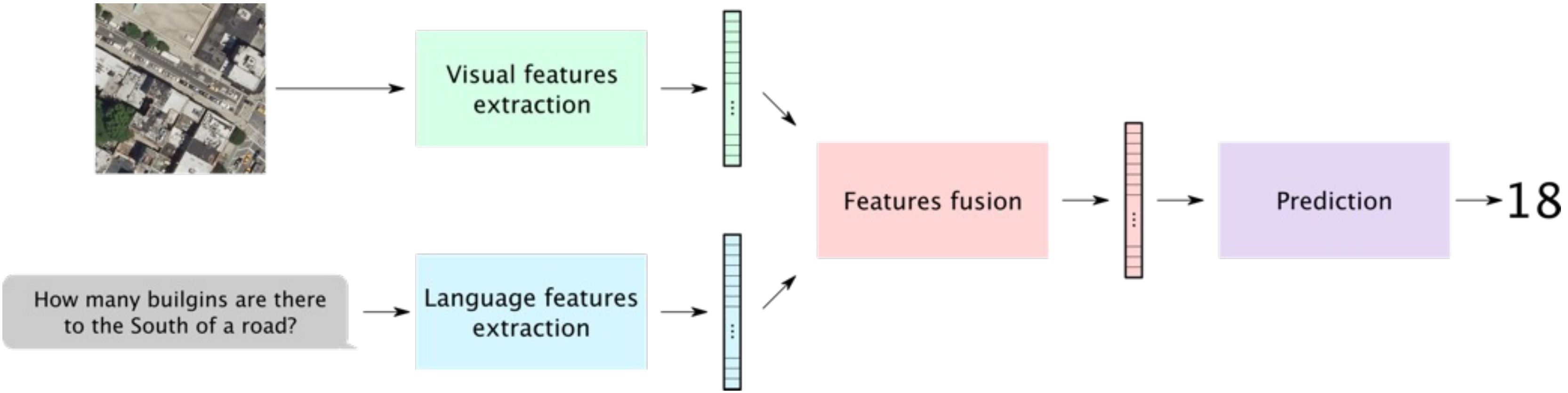
- Questions are not limited: can cover most of the computer vision tasks
- Can make computer vision results accessible to new people
- Limited applications today. Most studied one: answering question from visually impaired people

[1] Antol, Stanislaw, et al. "Vqa: Visual question answering."
Proceedings of the IEEE international conference on computer vision. 2015.

## Visual Question Answering

# Model

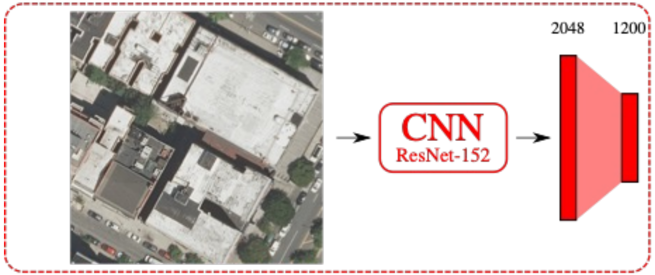
- Models look like this



## Visual Question Answering

# Model

A. Features extraction



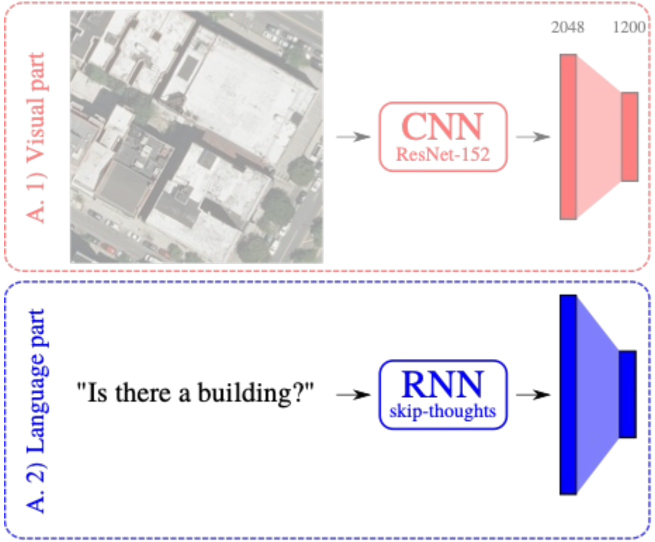


– ResNet-152 pre-trained on ImageNet with a fully-connected layer.

Lobry, Sylvain, et al. "RSVQA: Visual question answering for remote sensing data."
IEEE Transactions on Geoscience and Remote Sensing 58.12 (2020): 8555-8566.

## Visual Question Answering

# Model



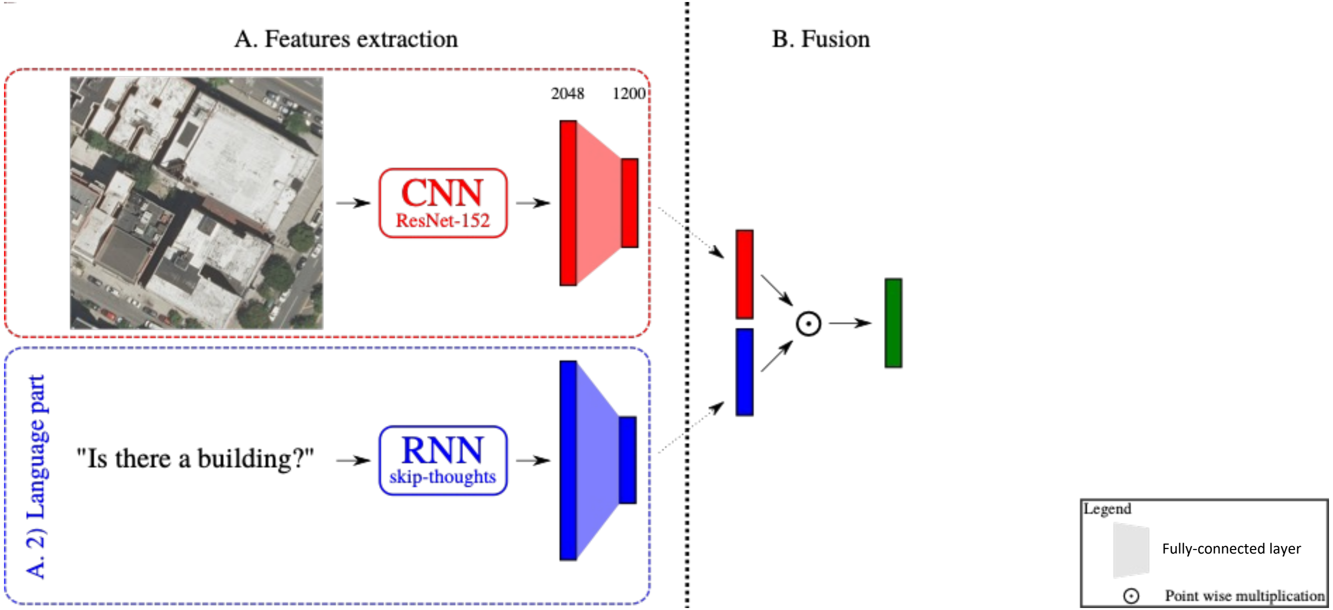


– Sequence encoder based on skip-thoughts : predict the previous and following sentences of a sentence in a book. Trained on BookCorpus.

Lobry, Sylvain, et al. "RSVQA: Visual question answering for remote sensing data." IEEE Transactions on Geoscience and Remote Sensing 58.12 (2020): 8555-8566.

Visual Question Answering

# Model

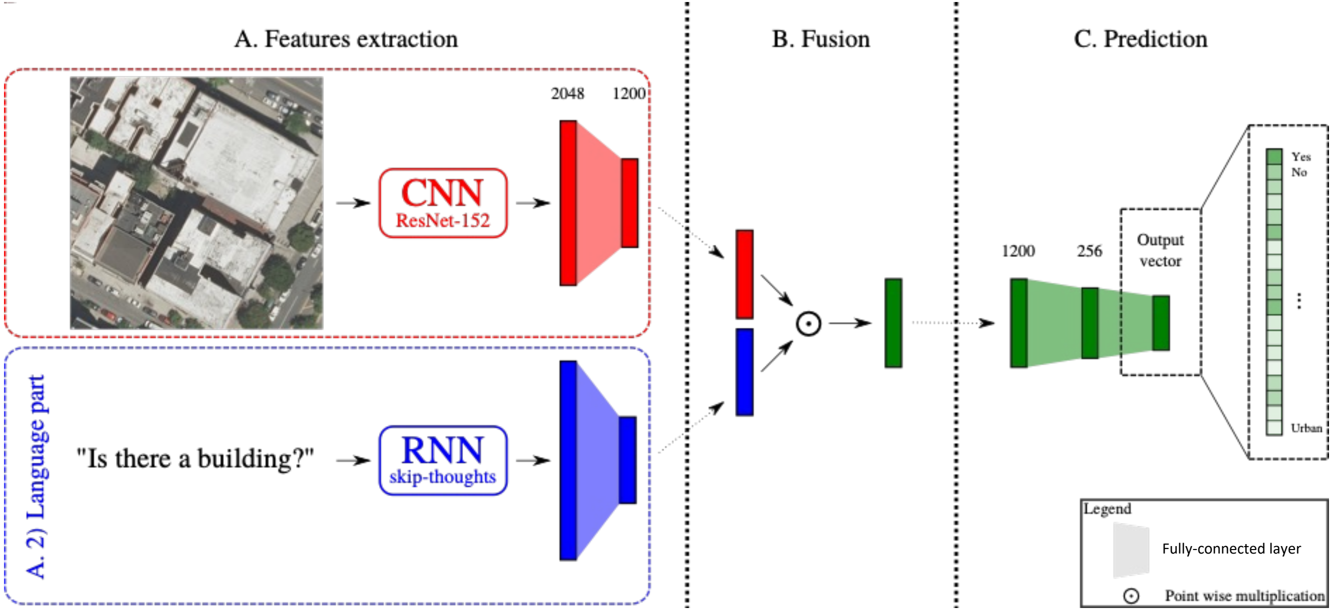



– Vectors fusion by point-wise multiplication.

Lobry, Sylvain, et al. "RSVQA: Visual question answering for remote sensing data." IEEE Transactions on Geoscience and Remote Sensing 58.12 (2020): 8555-8566.

Visual Question Answering

# Model

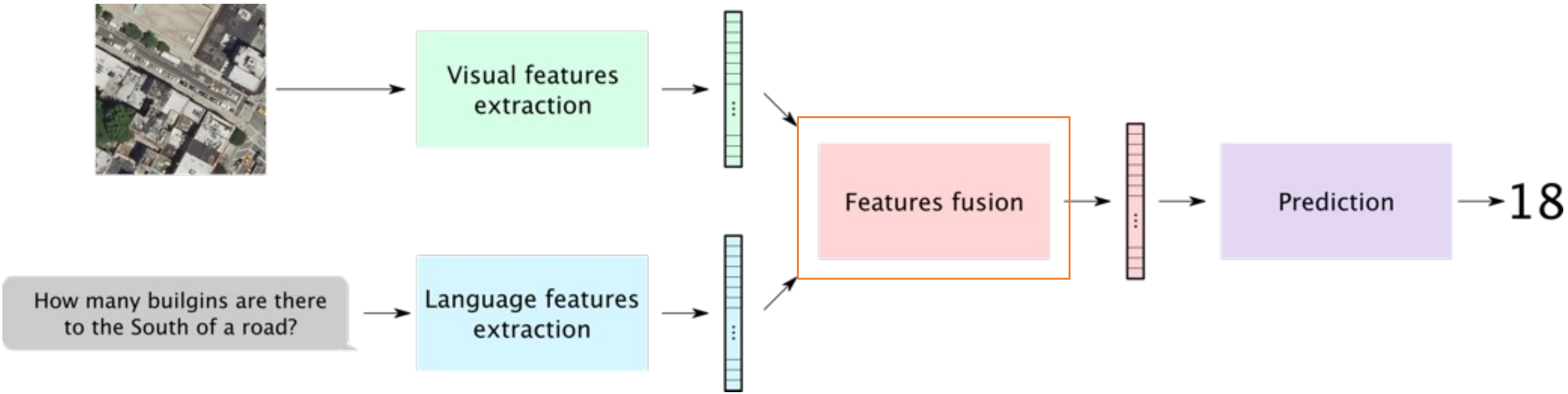



– Predict the most probable answer among a set of pre-defined ones.

Lobry, Sylvain, et al. "RSVQA: Visual question answering for remote sensing data." IEEE Transactions on Geoscience and Remote Sensing 58.12 (2020): 8555-8566.

Visual Question Answering

# Model



## Visual Question Answering

# Fusion

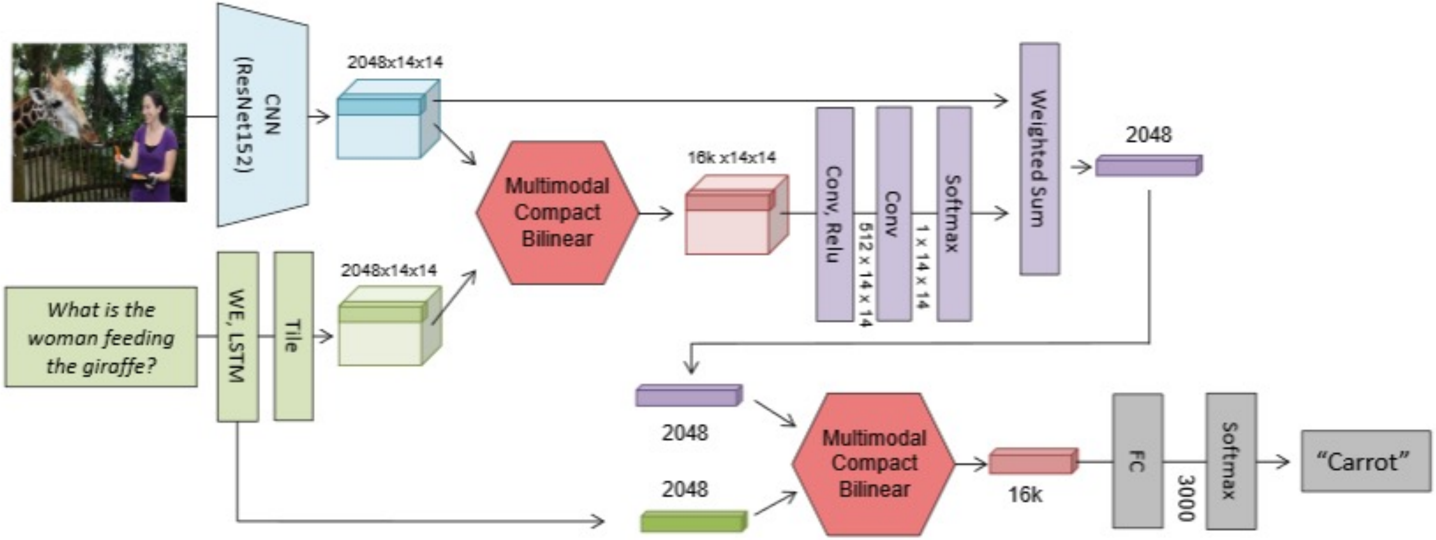
- Ideally we want to multiply both vectors
- Computationally intractable
- Possibility to use random projections [1], tensor decomposition [2]

[1] Fukui, Akira, et al. "Multimodal compact bilinear pooling for visual question answering and visual grounding." arXiv preprint arXiv:1606.01847 (2016).
[2] Ben-Younes, Hedi, et al. "Mutan: Multimodal tucker fusion for visual question answering." Proceedings of the IEEE international conference on computer vision. 2017.

Visual Question Answering

# Fusion – MCB model

- Fusion at two level: attention and prediction




[1] Fukui, Akira, et al. "Multimodal compact bilinear pooling for visual question answering and visual grounding." arXiv preprint arXiv:1606.01847 (2016).

A first simple solution

# Towards new application

**RSVQA LR**

772 images (Sentinel 2)

77’232 questions

Example:

What is the number of water areas? 7

**RSVQA HR**

10’659 images (USGS, 15cm)

955’664 questions

Example:

What is the amount of buildings? 7

**RSVQAxBEN**

590’326 images (Sentinel 2)

14’758’150 questions

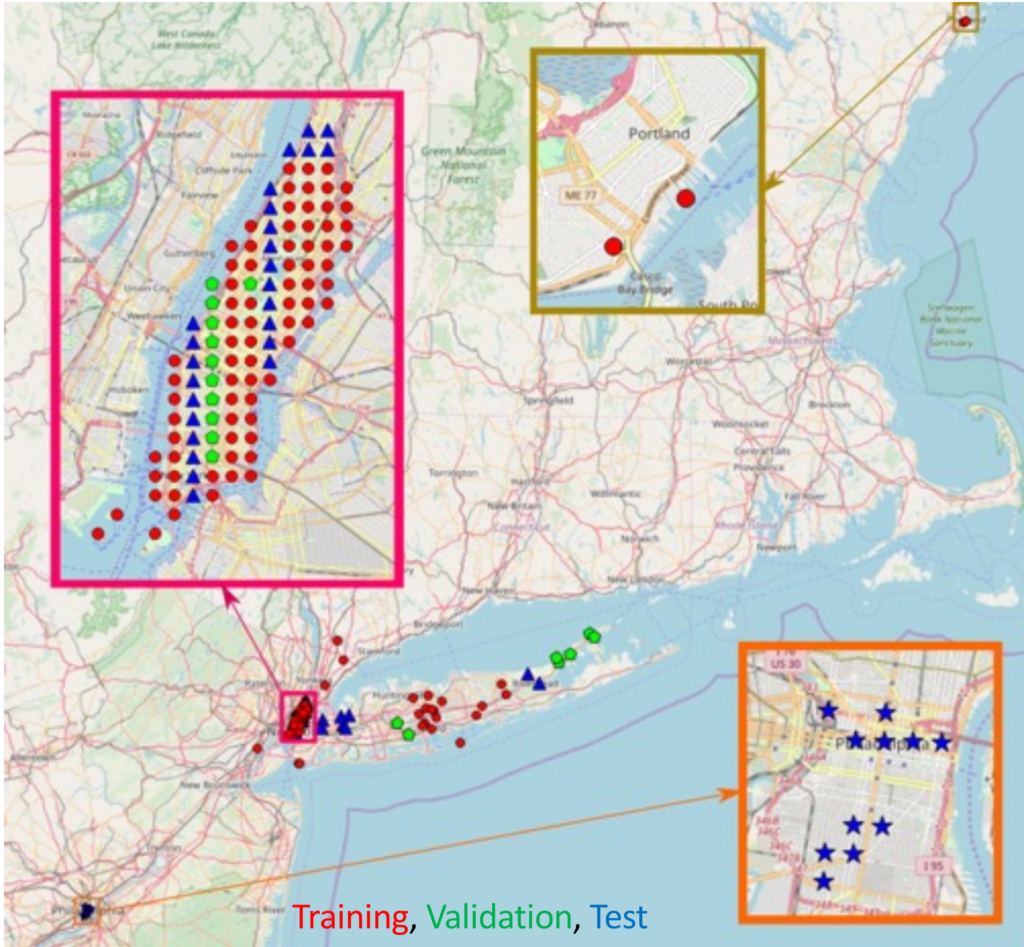
Example:

Are there artificial areas and agricultural areas or water bodies? Yes

## A first simple solution

# RSVQA HR

- East cost of the US
- 10’659 images from 161 orthophotos
- Two test sets to test spatial generalization.



## A first simple solution

# Results

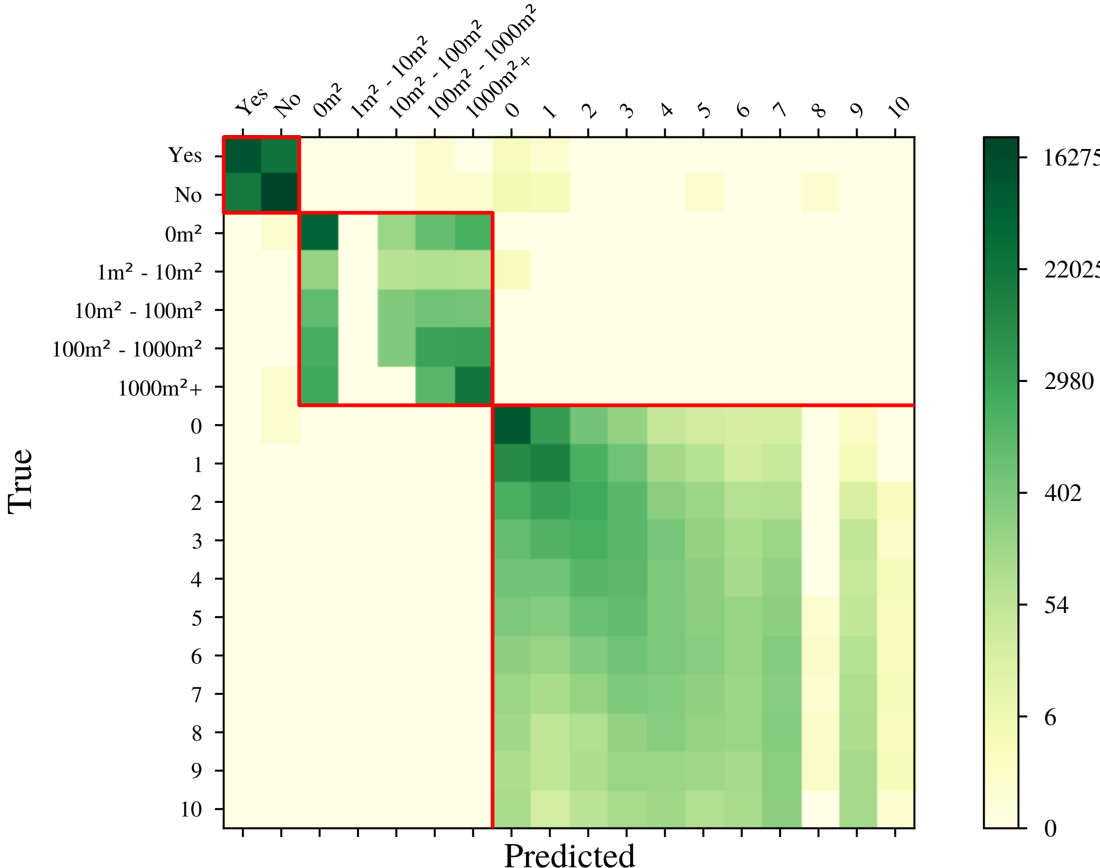
- Difficult to count
- Loss in performances on a new area

| Type | Accuracy Test set 1 | Accuracy Test set 2 |
|---|---|---|
| Count | 68.63% (0.11%) | 61.47% (0.08%) |
| Presence | 90.43% (0.04%) | 86.26% (0.47%) |
| Comparison | 88.19% (0.08%) | 85.94% (0.12%) |
| Area | 85.24% (0.05%) | 76.33% (0.50%) |
| AA | 83.12% (0.03%) | 77.50% (0.29%) |
| OA | 83.23% (0.02%) | 78.23% (0.25%) |

## A first simple solution

# Results

- Difficult to count
- Loss in performances on a new area
- When the answer is wrong, it is still logical



## A first simple solution

# Visual results

| Is there a residential building at the bottom of the place of worship? | |
|---|---|
| Ground truth | Prediction |
| yes | yes |

| What is the amount of large buildings? | |
|---|---|
| Ground truth | Prediction |
| 3 | 1 |

## Visual Question Answering

# Conclusion

- VQA is a new task
- Can open new usages of computer vision
- A very active research community on the topic