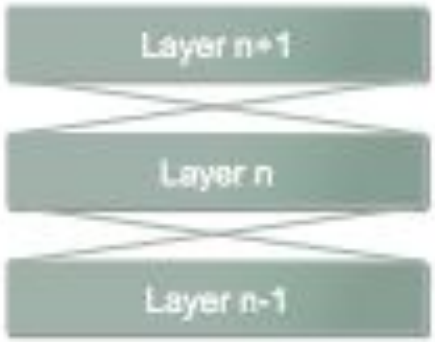
RECENT ADVANCES IN DNN

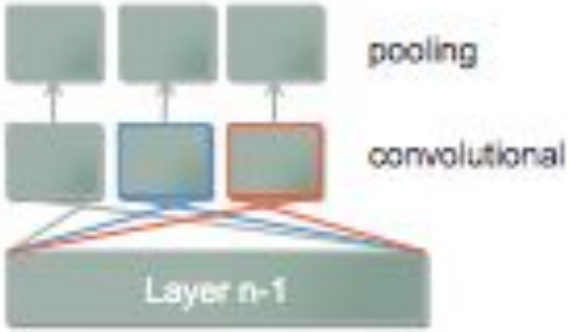
Neural networks

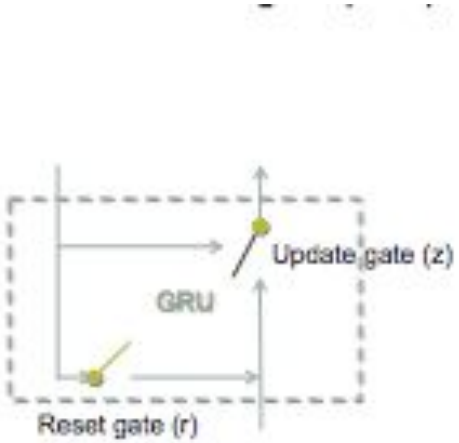
- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function and regularization
  - SGD and backprop
  - Learning rate
  - Overfitting – dropout, batchnorm

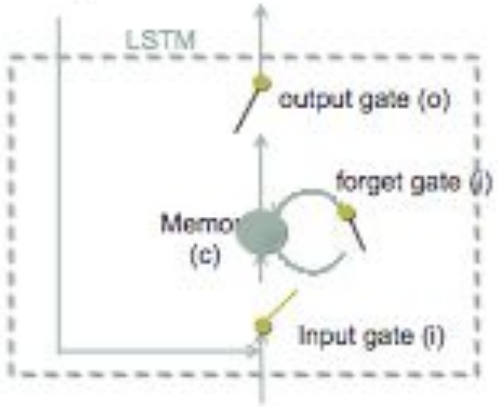
CNNs & RNNs

- CNNs
  - Matched filters
  - Convolution layer
  - Subsampling layer
    - Maxout, 1x1 convolution
- RNNs
  - Backpropagation Through time
  - GRUs (2 Gates)
  - LSTMs (3 Gates, explicit memory cell)









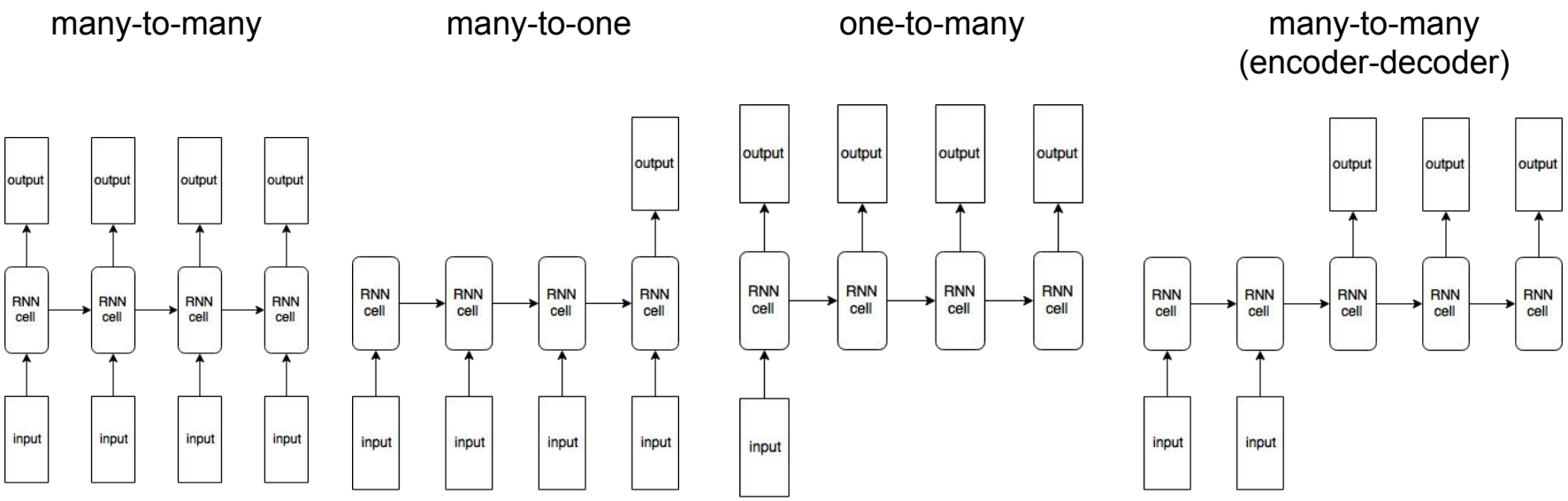
- You are not expected to understand any of the following parts in this lecture.
- This lecture serves as an overview of things going on with deep learning.
- If you are curious about any terms that gets thrown out there, see the relevant papers.

Misc topics

- Encoder-Decoder
- Attention modeling
- Transfer Learning, Multi-task models, Multi-models
- Object detection

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

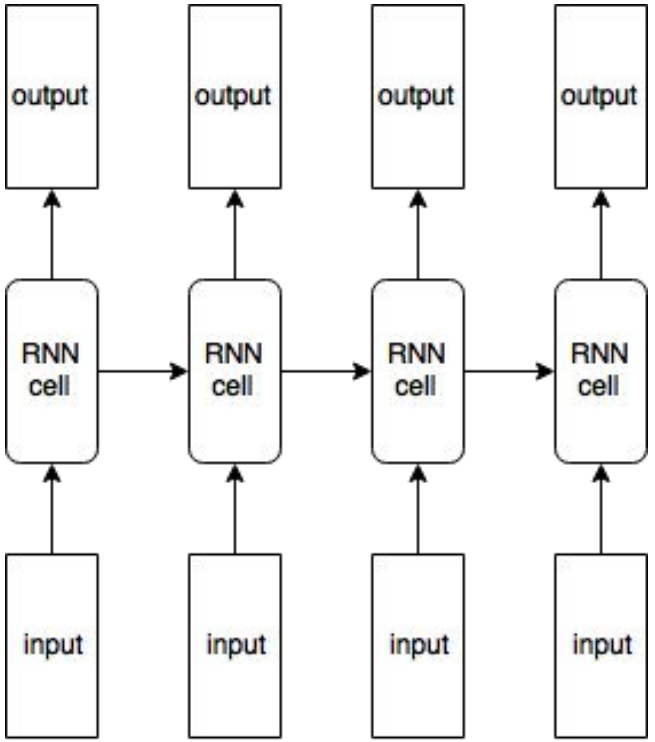
Different types of RNN architectures



Reference: The Unreasonable Effectiveness of Recurrent Neural Networks,
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

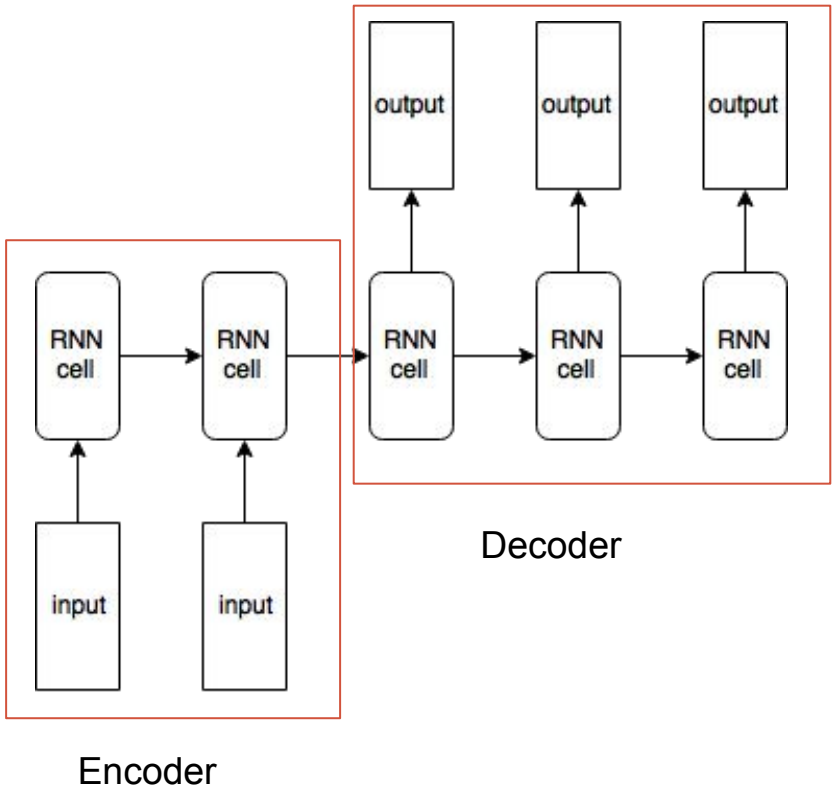
Many-to-many

- You have seen and implemented this type of RNN architecture in your homework (nowcasting)
- Sequence Input, Sequence Output



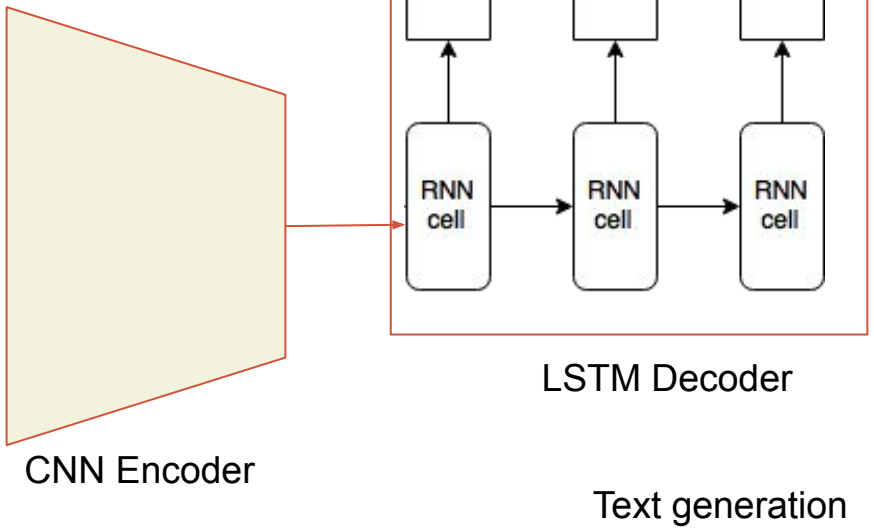
Many-to-many (encoder-decoder)

- Sequence Input, Sequence output
- These two sequences can be of different length
- E.g. Machine Translation
  - Input: English Sentence
  - Output: Thai Sentence



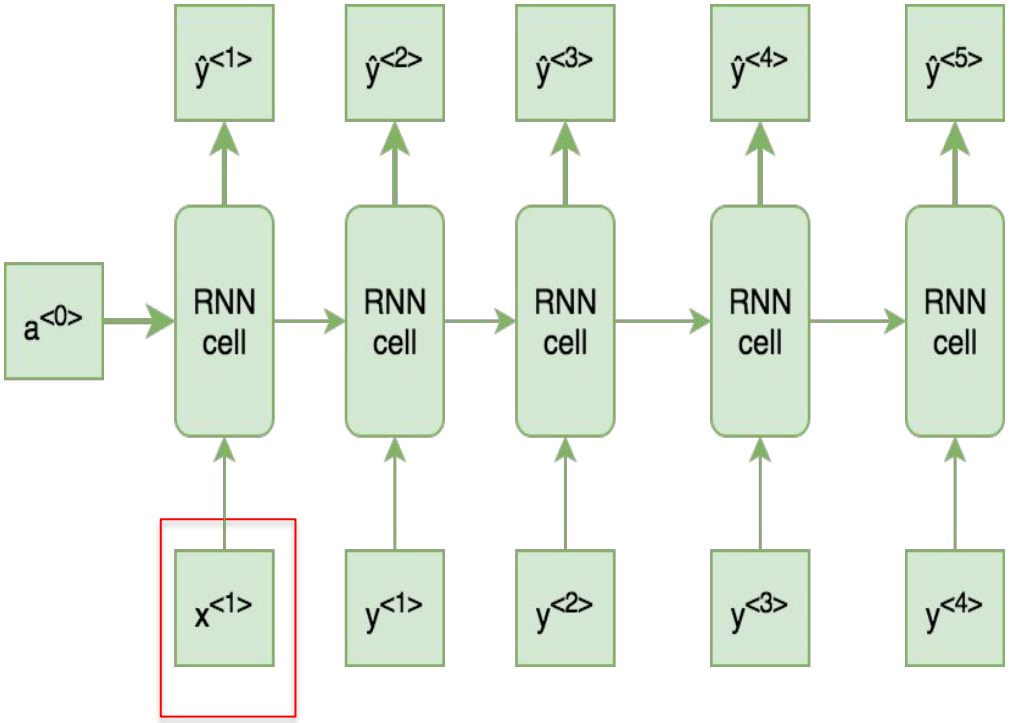
Encoder-decoder

- Input and output can be of different modality
  - Speech, text, image, etc.
- Speech to image (image generation)
- Image to text (image description)
- Image to code
  - https://sketch2code.azurewebsites.net/



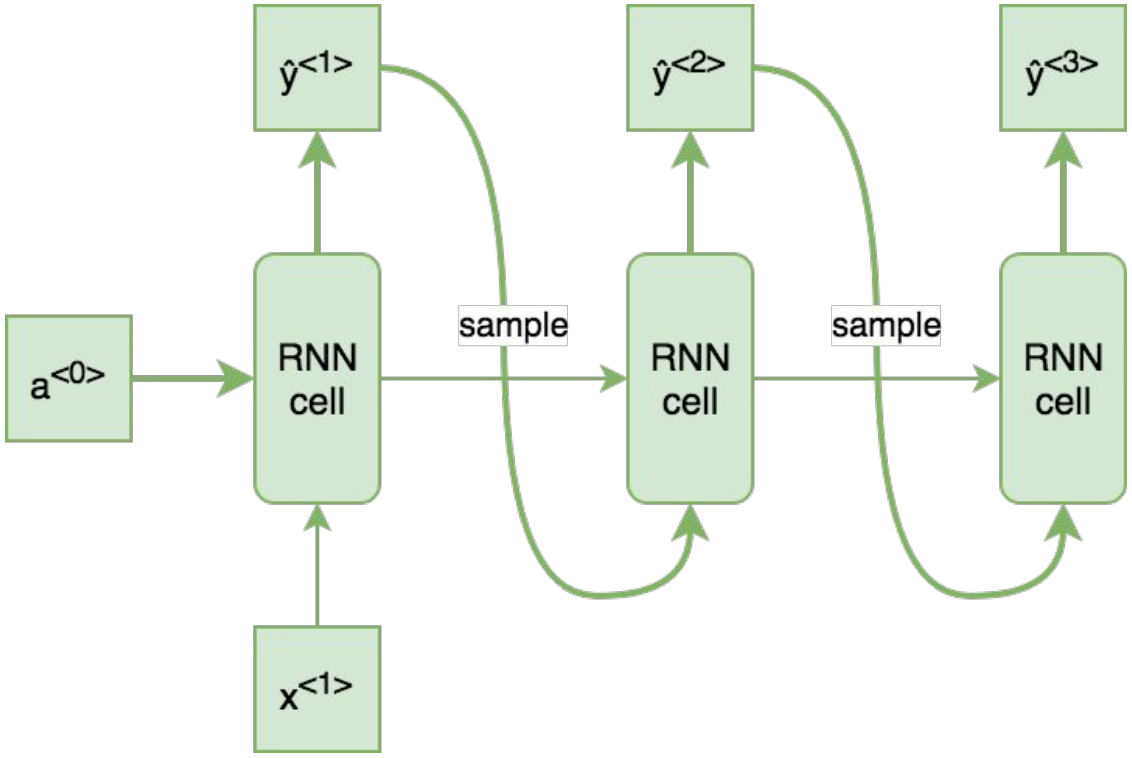
Text generation model (training)

- One-to-Many RNN
- The only real input is $x^{<1>}$
- $a^{<0>}$ is the initial hidden state.
- $\hat{y}$ is the predicted output.
- y is an actual output.
- During **the training phase**, instead of using the predicted output to feed into the next time-step, we use the actual output.



Text generation model (inference)

- To generate a novel sequence, the inference model (testing phase) randomly samples an output from a softmax distribution.



We are forced to pick one answer then proceed to the next step.
Auto-regressive decoding

Can reduce the error by doing beamsearch (try different possibilities but still pick at answer for each search space)

Text generation example

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

SEED: Democracy

> *Good morning. One of the borders will be able to continue to be here today. We have to say that the partnership was a partnership with the American people and the street continually progress that is a process and distant lasting peace and support that they were supporting the work of concern in the world. They were in the streets and communities that could have to provide steps to the people of the United States and Afghanistan. In the streets—the final decade of the country that will include the people of the United States of America. Now, humanitarian crisis has already rightly achieved the first American future in the same financial crisis that they can find reason to invest in the world.*
>
> *Thank you very much. God bless you. God bless you. Thank you.*

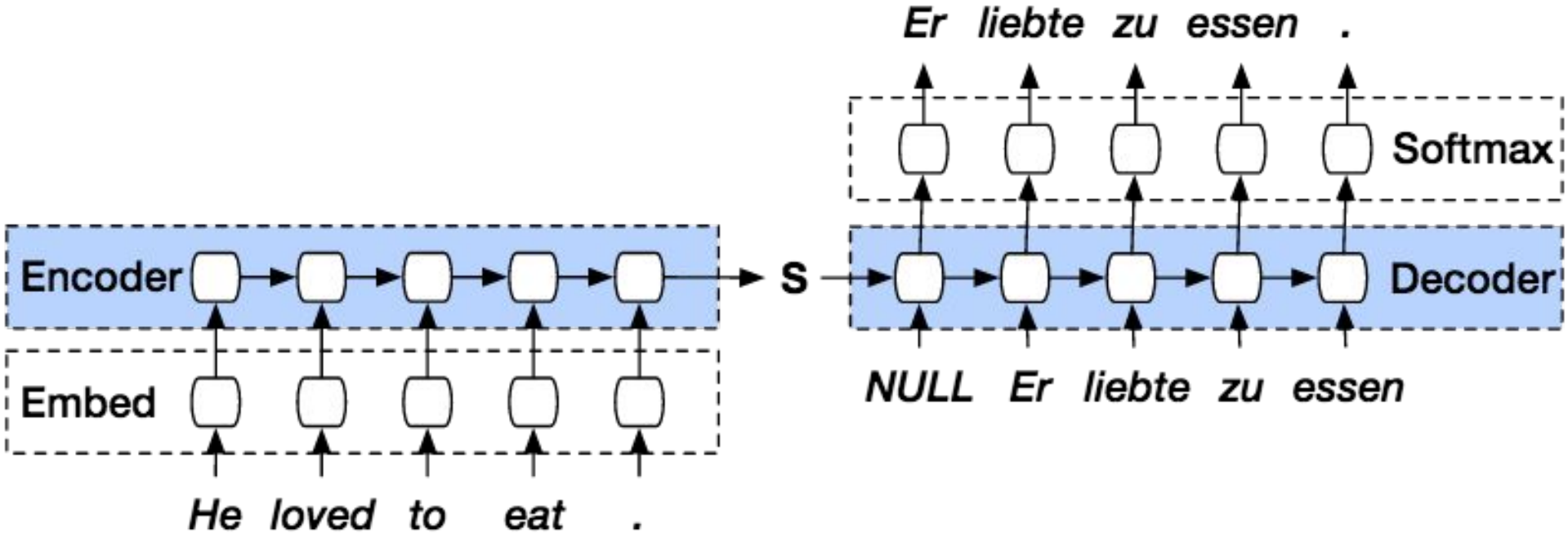
Music generation

Input: initial note
Predict: note differences and chord

https://www.researchgate.net/publication/327811979_Algorithmic_Music_Composition_Comparison

Machine translation using encoder-decoder

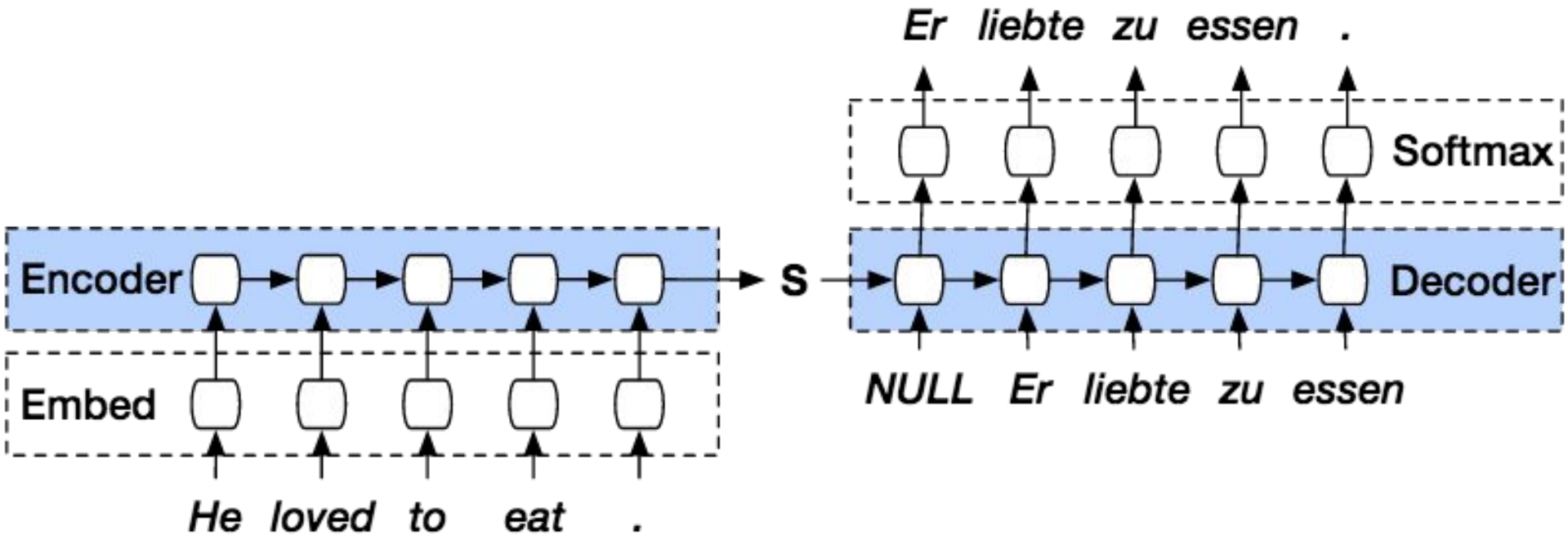
- Use another LSTM as the decoder
- Input to the LSTM is the encoded sentence and the previously output word



Ilya Sutskever, Sequence to Sequence Learning with Neural Networks, https://arxiv.org/abs/1409.3215

Problems with LSTM/GRU

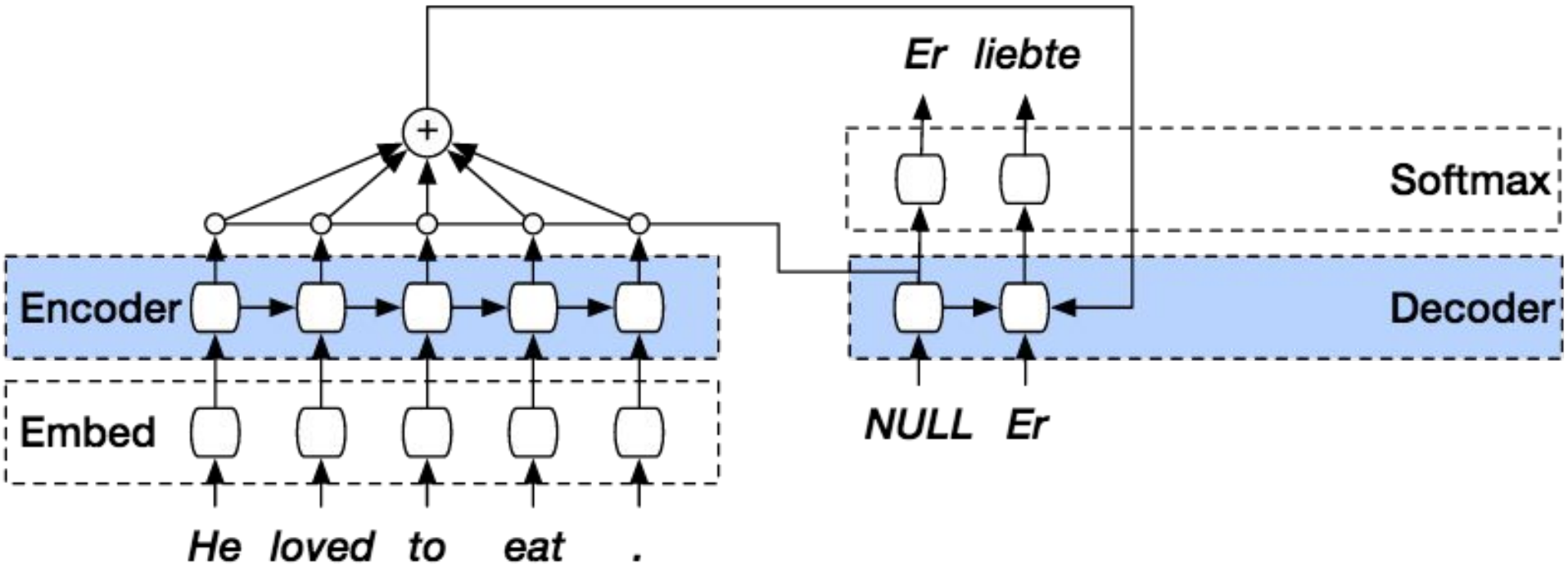
- Cannot handle long sequence well (limited fixed size memory S)
  - Also vanishing gradients



Ilya Sutskever, Sequence to Sequence Learning with Neural Networks, https://arxiv.org/abs/1409.3215

Attention Mechanism

Attention allows the decoder part of the network to focus/**attend** on a different part of the encoder's outputs for every step of the decoder's own outputs.



Dzmitry Bahdanau, Neural Machine Translation by Jointly Learning to Align and Translate, https://arxiv.org/abs/1409.0473

Attention Mechanism

$$\mathbf{c}_i = \sum_j a_{ij} \mathbf{h}_j$$

Encoder time step

Decoder time step

attention score

context vector

previous hidden state/ encoder state

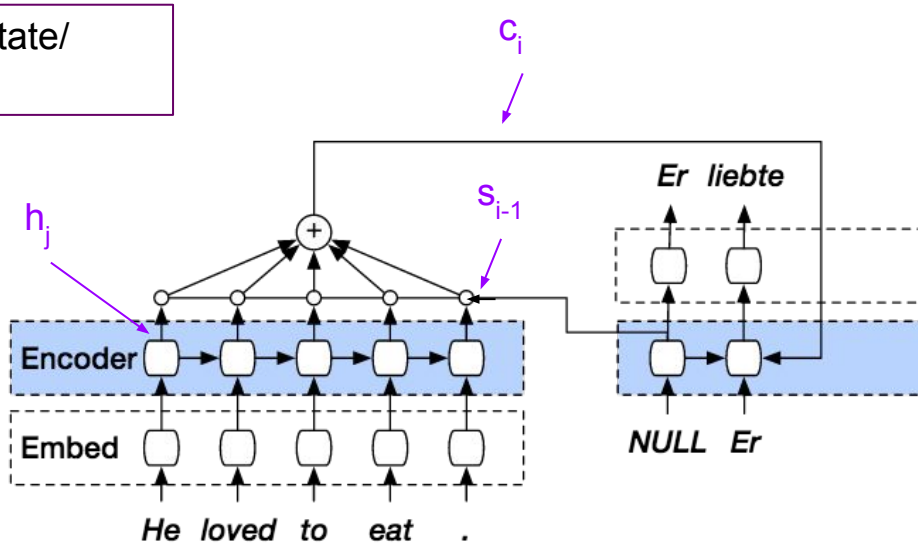
We want to calculate a context vector ***c*** based on hidden states $\mathbf{s}_0 \ldots \mathbf{s}_{m-1}$ that can be used with the current state $\mathbf{h}_j$ for prediction. The context vector $\mathbf{c}_i$ at position "i" is calculated as an average of the previous states weighted with the attention scores $\mathbf{a}_i$.

$$a_i = softmax(f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j))$$

attention scores

current hidden state/ decoder state

previous hidden state/ encoder state



Reference:http://ruder.io/deep-learning-nlp-best-practices/index.html#attention

Attention Mechanism

Encoder time step

$$\mathbf{c}_i = \sum_j a_{ij} \mathbf{h}_j$$

Weighted sum of hidden states
Softmax ensues summing to 1

Decoder time step

attention score

context vector

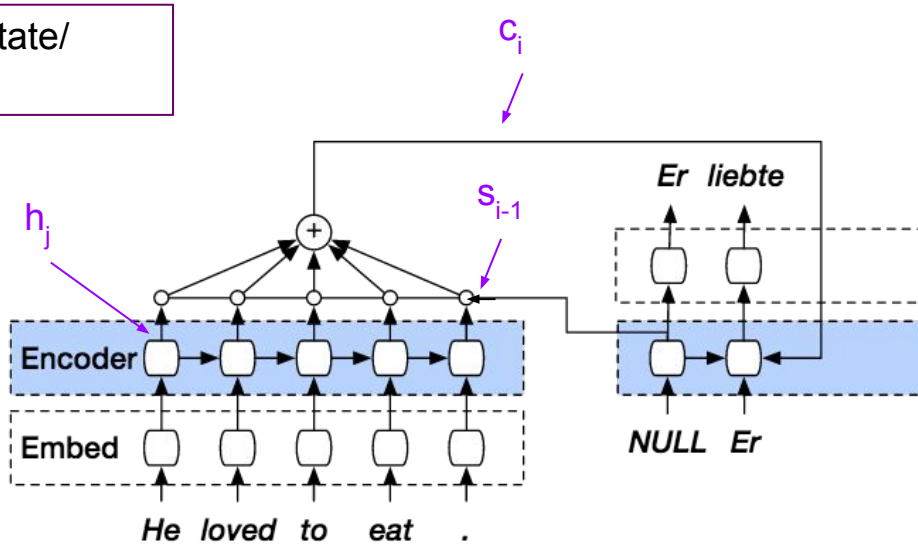
previous hidden state/ encoder state

$$a_i = softmax(f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j))$$

attention scores

current hidden state/ decoder state

previous hidden state/ encoder state



Attention Mechanism

Encoder time step

$$\mathbf{c}_i = \sum_j a_{ij} \mathbf{h}_j$$

Decoder time step

attention score

context vector

previous hidden state/ encoder state

f_{att} is some function that takes in s and h

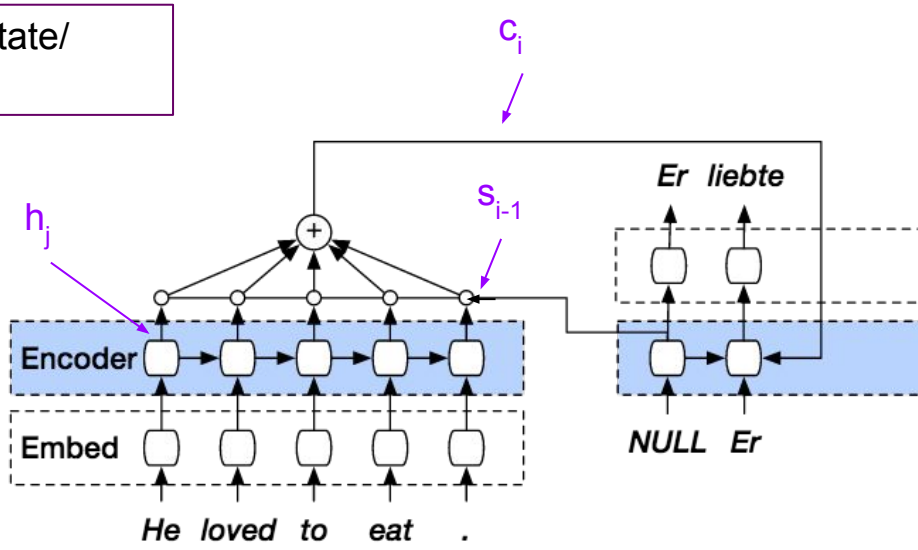
f_{att} typically is a dense layer with s and h as inputs (concatenate)

$$a_i = softmax(f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j))$$

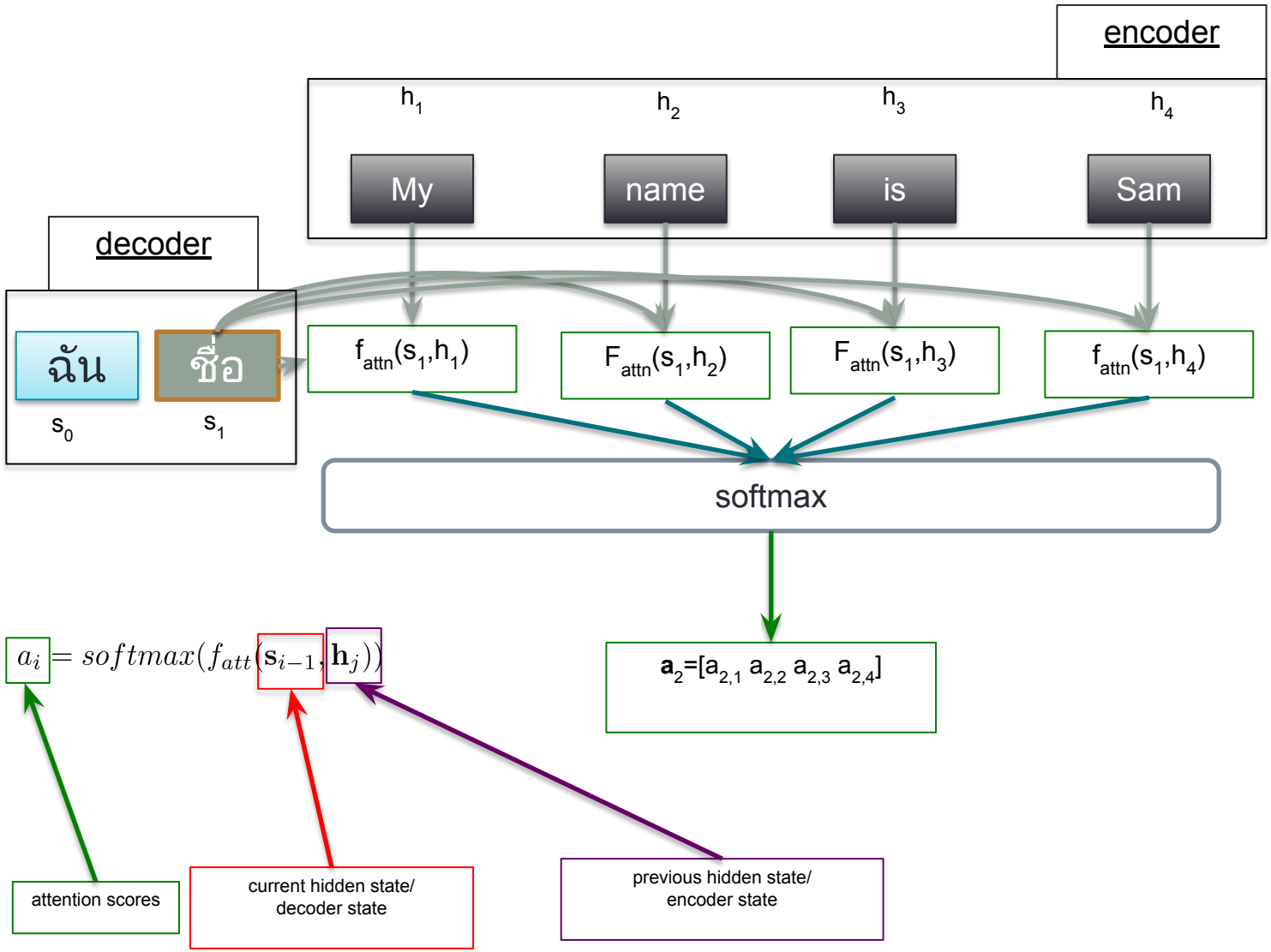
attention scores

current hidden state/ decoder state

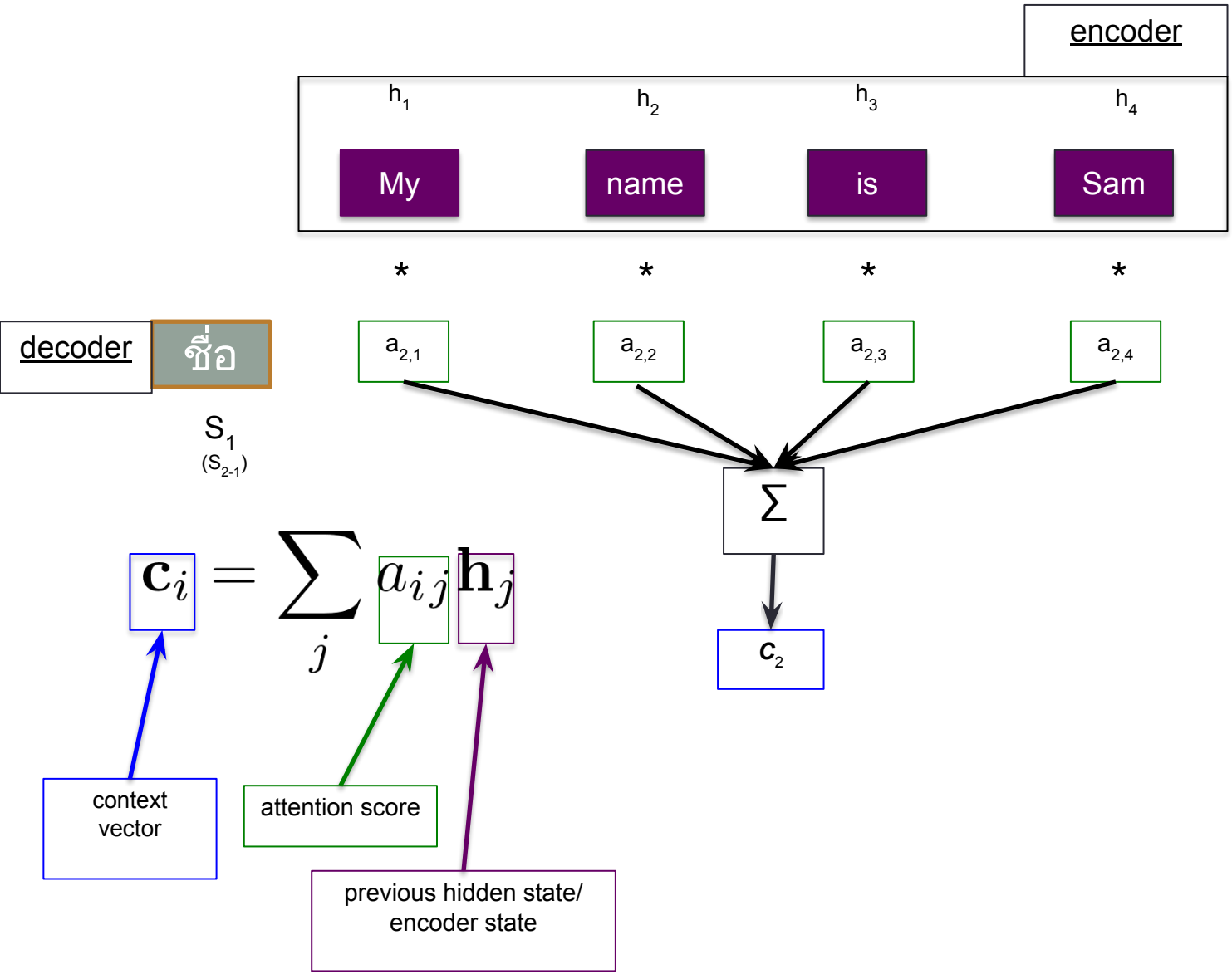
previous hidden state/ encoder state



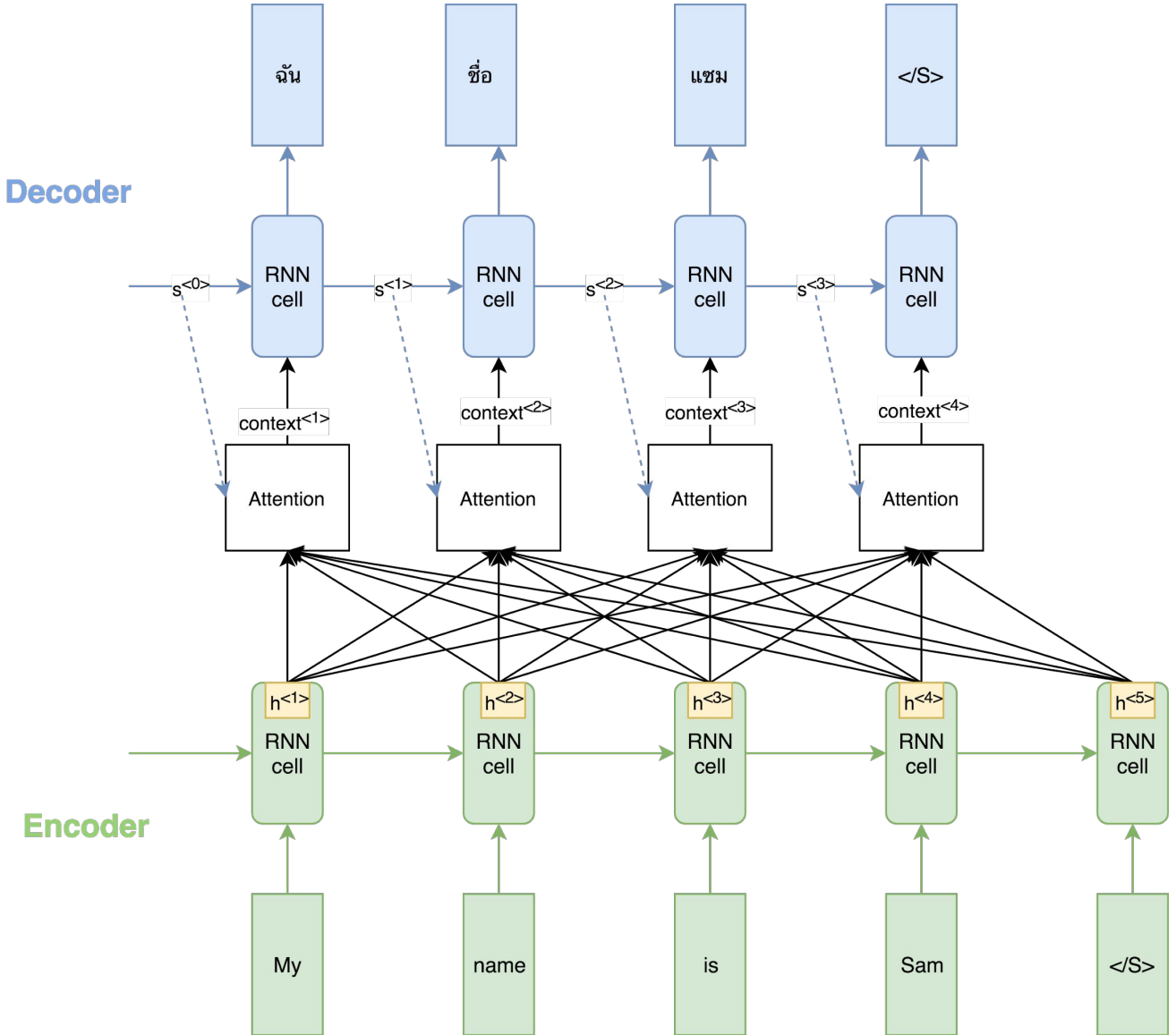
Attention Calculation Example



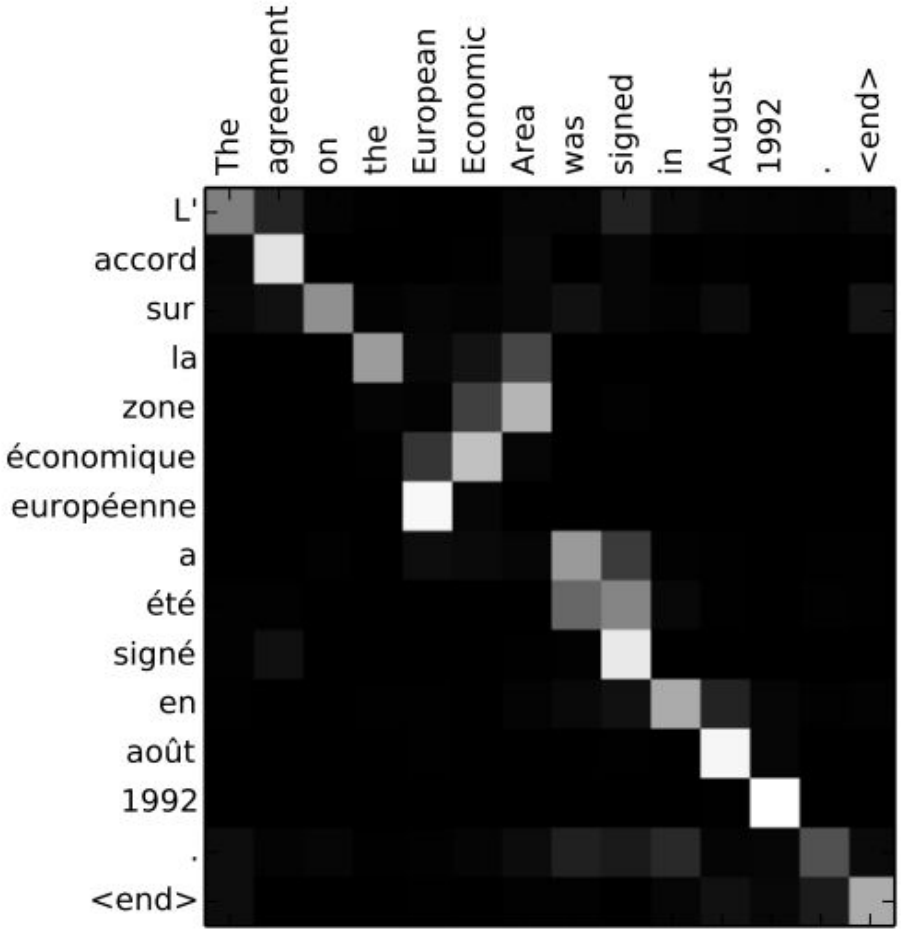
Attention Calculation Example



RNN and attention mechanism



Graphical Example: English-to-French machine translation



Reference: Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR(2015).

Type of Attention mechanisms

There are many variants of attention function f_{attn}

Additive attention: The original attention mechanism (Bahdanau et al., 2015) uses a one-hidden layer feed-forward network to calculate the attention alignment:

$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = tanh(\mathbf{W}_a[\mathbf{s}_{i-1}; \mathbf{h}_j])$$

Multiplicative attention: Multiplicative attention (Luong et al., 2015) simplifies the attention operation by calculating the following function:

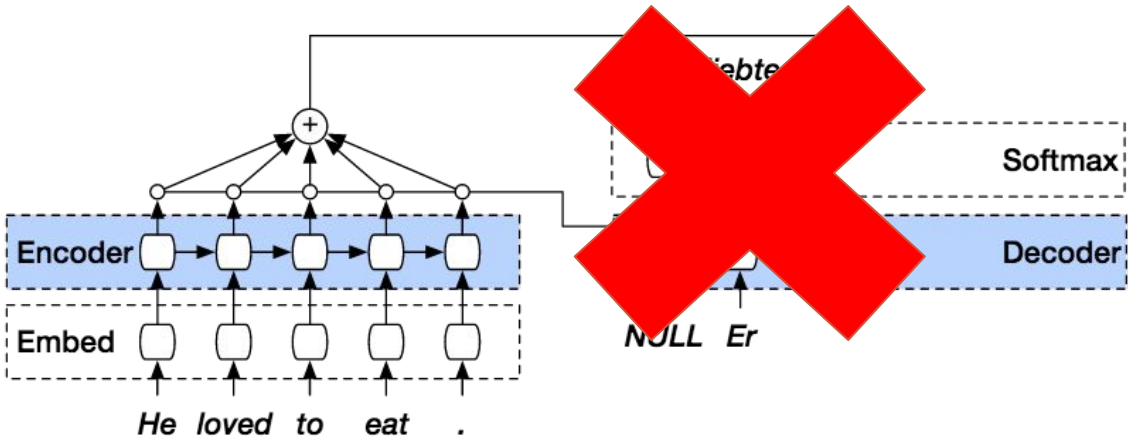
$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{s}_{i-1}^{\top} \mathbf{W}_a \mathbf{h}_j$$

Self-attention: Without any additional information, however, we can still extract relevant aspects from the sentence by allowing it to attend to itself using self-attention (Lin et al., 2017)

$$\mathbf{a} = softmax(\mathbf{w}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T))$$

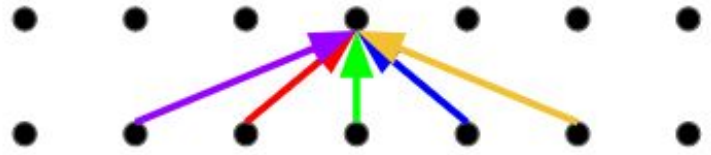
No s from encoder (Encoder side can also self-attend)

Self-attention



No need for additional information in order to select where to attend

Convolution



Similar to CNN in flow

Self-Attention

https://nlp.stanford.edu/seminar/details/lkaiser.pdf

Self-attention example

- if I can give this restaurant a 0 I will we be just ask our waitress leave because someone with a reservation be wait for our table my father and father-in-law be still finish up their coffee and we have not yet finish our dessert I have never be so humiliated do not go to this restaurant their food be mediocre at best if you want excellent Italian in a small intimate restaurant go to dish on the South Side I will not be go back
- this place suck the food be gross and taste like grease I will never go here again ever sure the entrance look cool and the waiter can be very nice but the food simply be gross taste like cheap 99cent food do not go here the food shot out of me quick then it go in
- everything be pre cook and dry its crazy most Filipino people be used to very cheap ingredient and they do not know quality the food be disgusting I have eat at least 20 different Filipino family home this not even mediocre
- seriously f *** this place disgust food and shitty service ambience be great if you like dine in a hot cellar engulf in stagnate air truly it be over rate over price and they just under deliver forget try order a drink here it will take forever get and when it finally do arrive you will be ready pass out from heat exhaustion and lack of oxygen how be that a head change you do not even have pay for it I will not disgust you with the detailed review of everything I have try here but make it simple it all suck and after you get the bill you will be walk out with a sore ass save your money and spare your self the disappointment
- i be so angry about my horrible experience at Medusa today my previous visit be amaze 5/5 however my go to out of town and I land an appointment with Stephanie I go in with a picture of roughly what I want and come out look absolutely nothing like it my hair be a horrible ashy blonde not anywhere close to the platinum blonde I request she will not do any of the pop of colour I want and even after specifically tell her I do not like blunt cut my hair have lot of straight edge she do not listen to a single thing I want and when I tell her I be unhappy with the colour she basically tell me I be wrong and I have do it this way no no I do not if I can go from Little Mermaid red to golden blonde in 1 sitting that leave my hair fine I shall be able go from golden blonde to a shade of platinum blonde in 1 sitting thanks for ruin my New Year's with 1 the bad hair job I have ever have

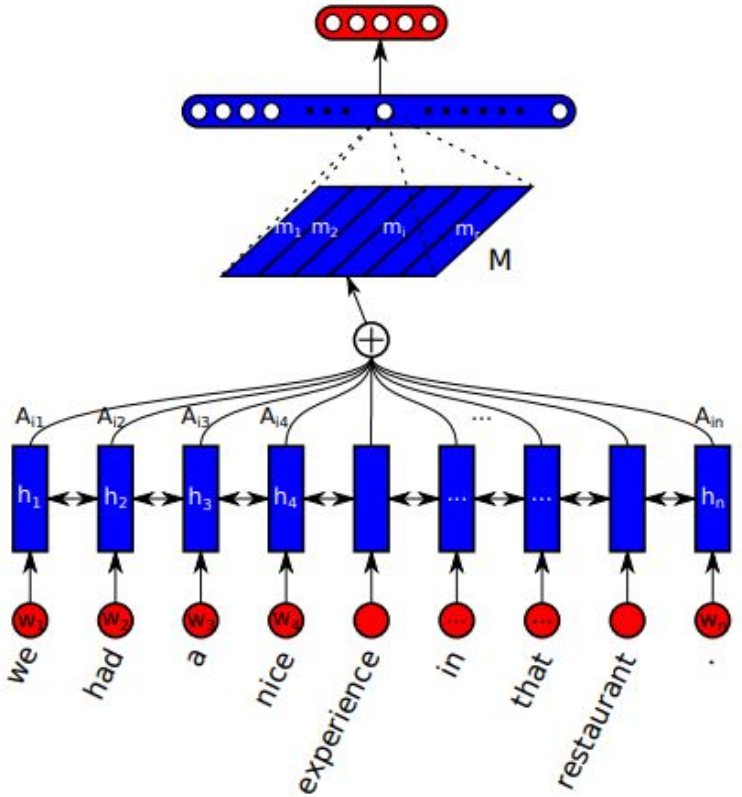
(a) 1 star reviews

- I really enjoy Ashley and Ami salon she do a great job be friendly and professional I usually get my hair do when I go to MI because of the quality of the highlight and the price the price be very affordable the highlight fantastic thank Ashley i highly recommend you and ill be back
- love this place it really be my favorite restaurant in Charlotte they use charcoal for their grill and you can taste it steak with chimichurri be always perfect Fried yucca cilantro rice pork sandwich and the good tres lech I have had.The desert be all incredible if you do not like it you be a mutant if you will like diabeetus try the Inca Cola
- this place be so much fun I have never go at night because it seem a little too busy for my taste but that just prove how great this restaurant be they have amazing food and the staff definitely remember us every time we be in town I love when a waitress or waiter come over and ask if you want the cab or the Pinot even when there be a rush and the staff be run around like crazy whenever I grab someone they instantly smile acknowlegde us the food be also killer I love when everyone know the special and can tell you they have try them all and what they pair well with this be a first last stop whenever we be in Charlotte and I highly recommend them
- great food and good service .... what else can you ask for everything that I have ever try here have be great
- first off I hardly remember waiter name because its rare you have an unforgettable experience the day I go I be celebrate my birthday and let me say I leave feel extra special our waiter be the best ever Carlos and the staff as well I be with a party of 4 and we order the potato salad shrimp cocktail lobster amongst other thing and boy be the food great the lobster be the good lobster I have ever eat if you eat a dessert I will recommend the cheese cake that be also the good I have ever have it be expensive but so worth every penny I will definitely be back there go again for the second time in a week and it be even good ..... this place be amazing

(b) 5 star reviews

Figure 2: Heatmap of Yelp reviews with the two extreme score.

Red means word is often attended to



https://arxiv.org/pdf/1703.03130.pdf

Type of Attention mechanisms

There are many variants of attention function f_{attn}

Additive attention: The original attention mechanism (Bahdanau et al., 2015) uses a one-hidden layer feed-forward network to calculate the attention alignment:

$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = tanh(\mathbf{W}_a[\mathbf{s}_{i-1}; \mathbf{h}_j])$$

Multiplicative attention: Multiplicative attention (Luong et al., 2015) simplifies the attention operation by calculating the following function:

$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{s}_{i-1}^{\top} \mathbf{W}_a \mathbf{h}_j$$

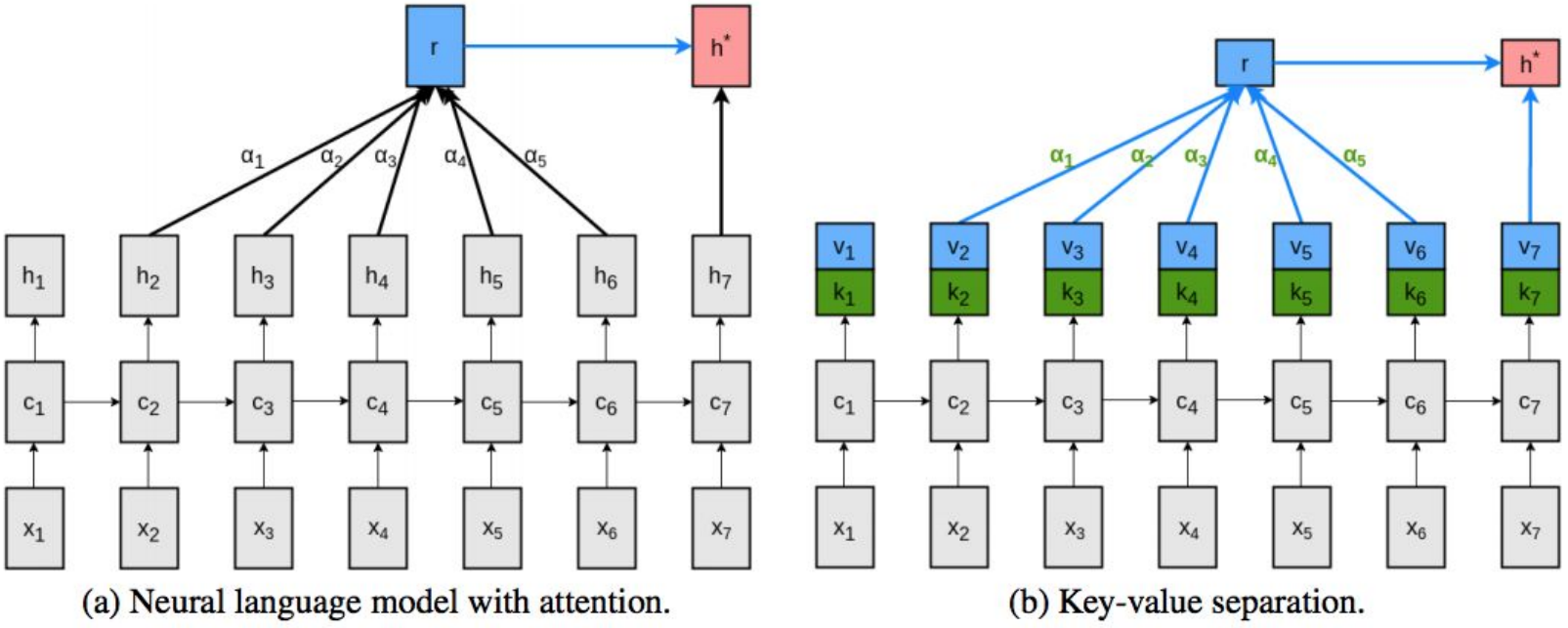
Self-attention: Without any additional information, however, we can still extract relevant aspects from the sentence by allowing it to attend to itself using self-attention (Lin et al., 2017)

$$\mathbf{a} = softmax(\mathbf{w}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T))$$

No s from encoder (Encoder side can also self-attend)

Key-value attention:key-value attention (Daniluk et al., 2017) is a recent attention variant that separates form from function by keeping separate vectors for the attention calculation.

Key-value attention



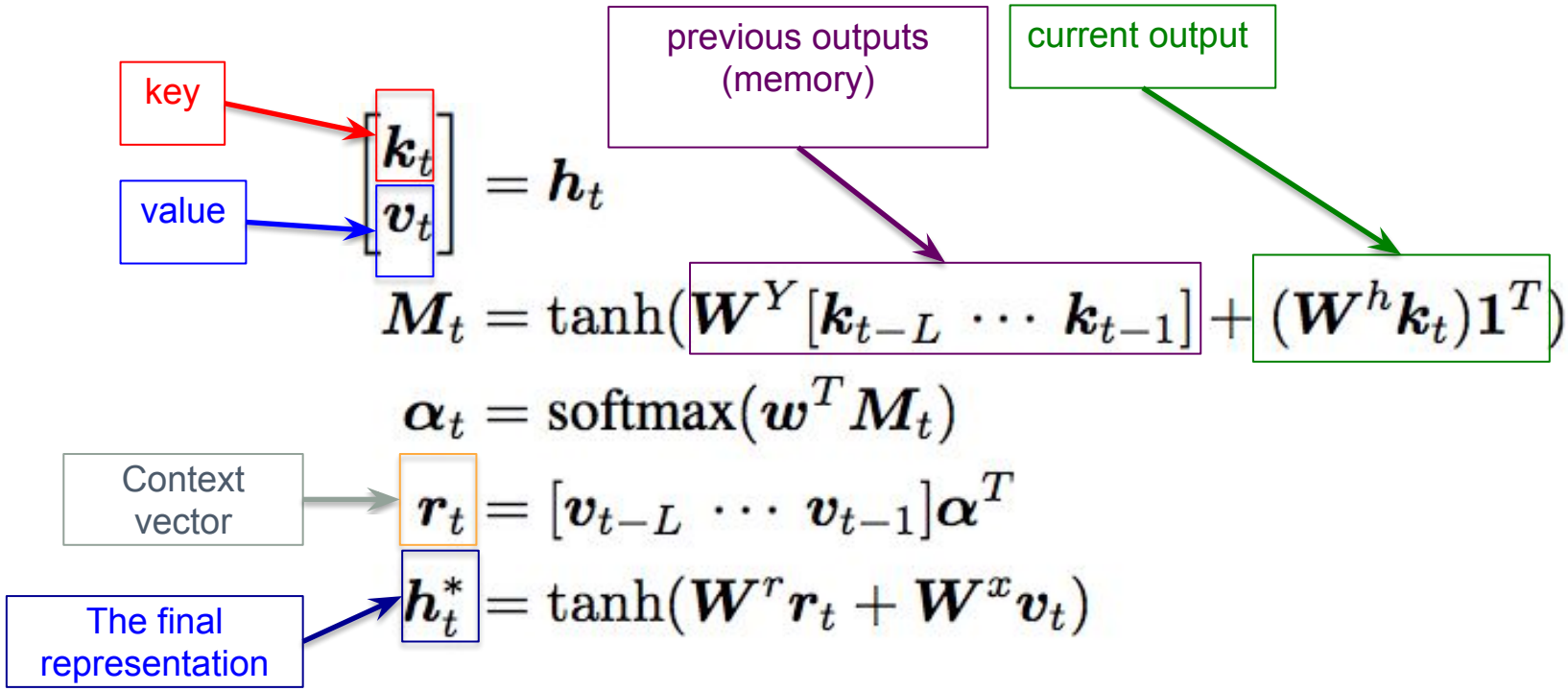
(a) Neural language model with attention.

(b) Key-value separation.

K (keys) for finding attention weights
V (value) for using as features to sum

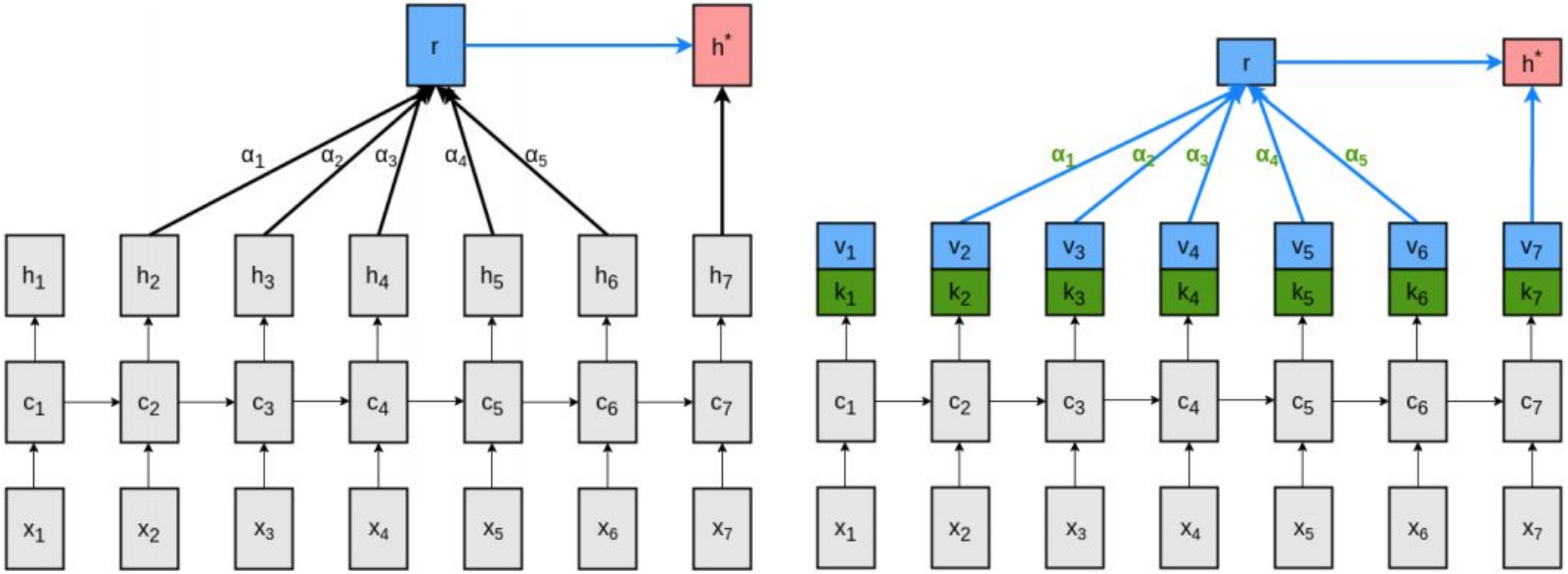
Reference: Daniluk, M., Rockt, T., Welbl, J., & Riedel, S. (2017). Frustratingly Short Attention Spans in Neural Language Modeling. In ICLR 2017.

Key-value attention



Problems with Recurrent networks

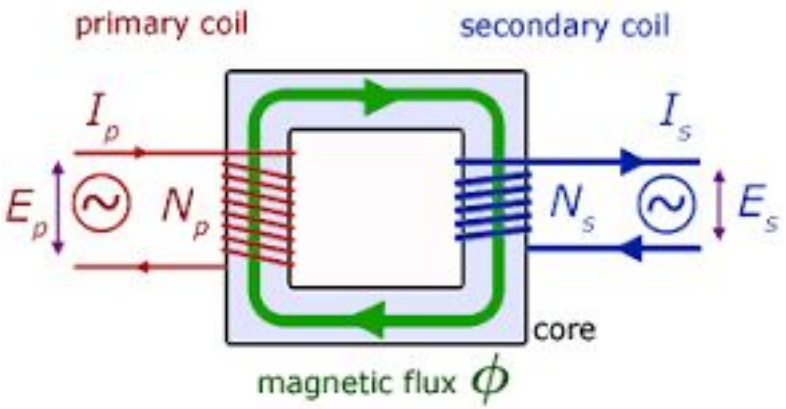
Slow. Feed forward needs to wait for output from previous output



(a) Neural language model with attention.

(b) Key-value separation.

Transformer?



Attention is all you need

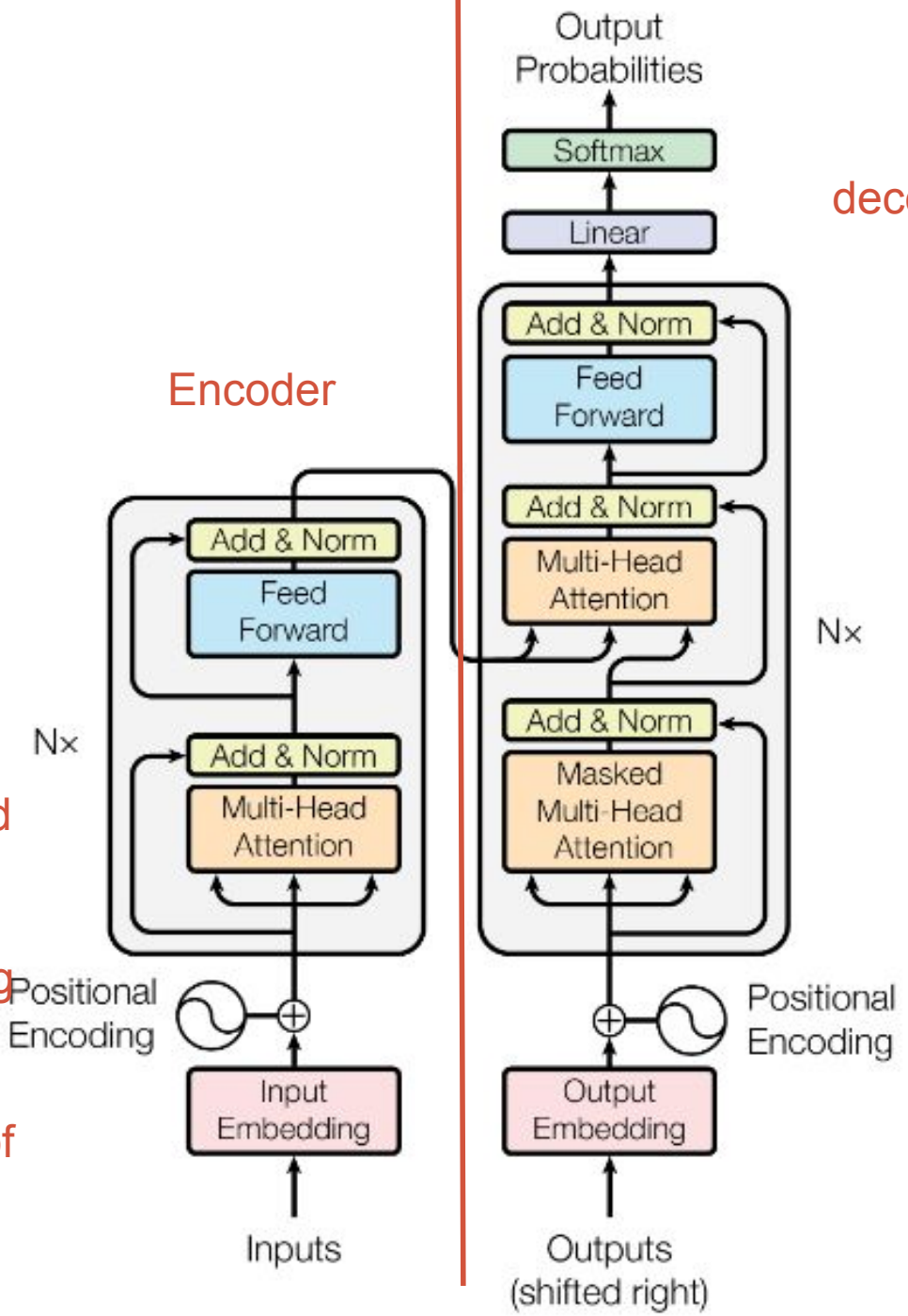
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

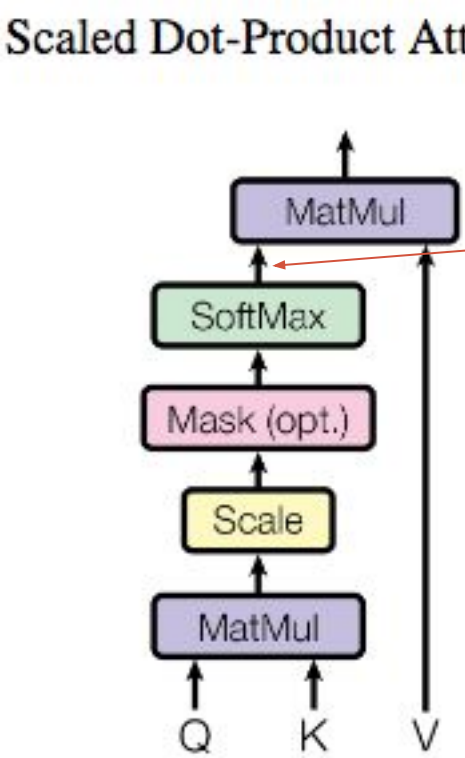
https://arxiv.org/pdf/1706.03762.pdf

To eliminate GRU which remembers time, encode position instead

Done by adding some fixed number for each location of the word in the sentence



Key Value Query attention



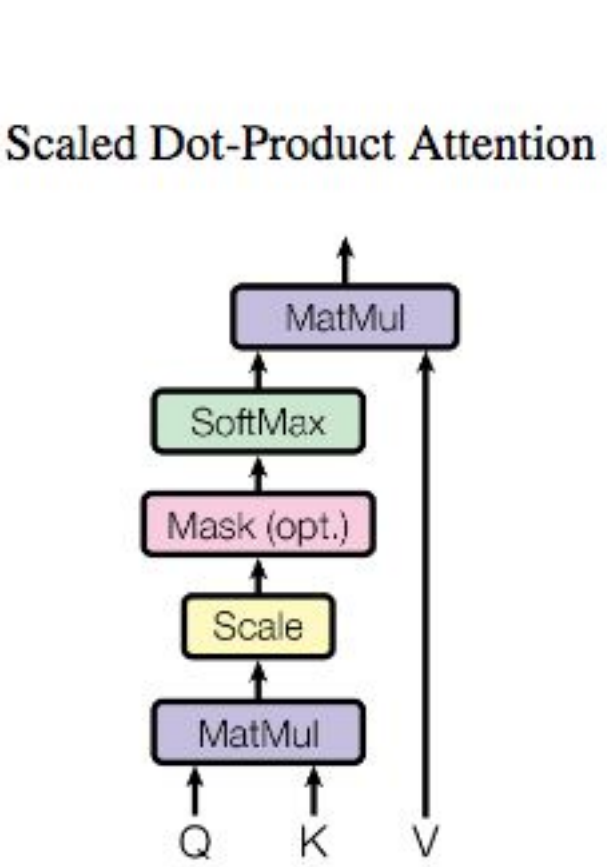
Attention weights

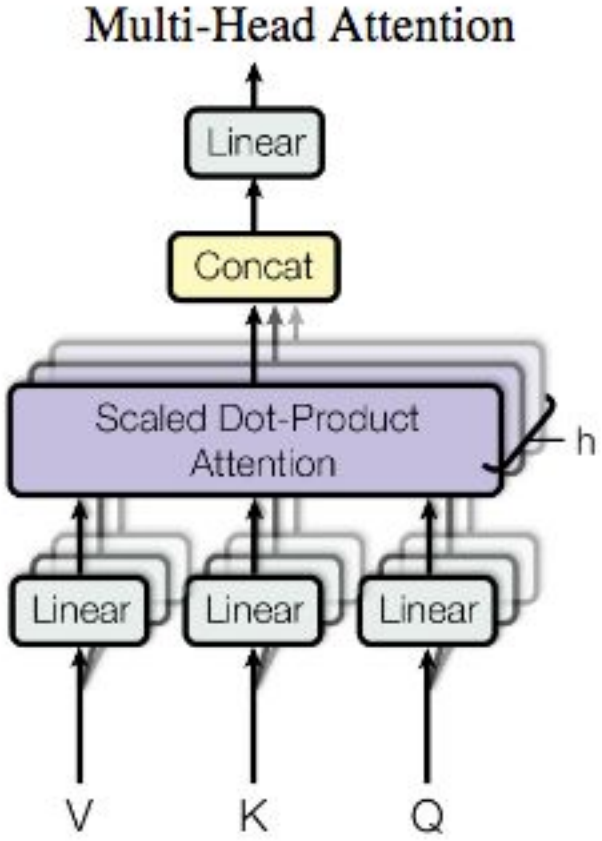
Query – used with Key to determine the position to attend

Value – used as the information after determining the position

Machine Translation: K, V = Encoder side
Q = Decoder side
Self attention: Q, K, V = Encoder side

Multi-head attention





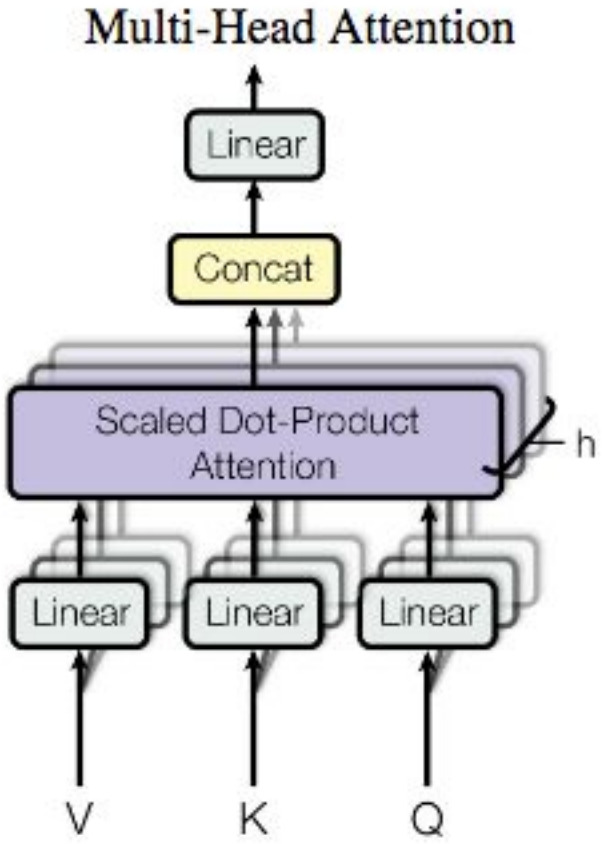
Query – used with Key to determine the position
Value – used as the information after determining the position

Attention drawback

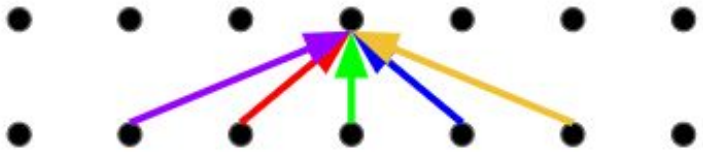
- Convolution: weights * input. Each weights are different. So position is encoded.
- Self-attention: a weighted average. Position information is lost at the output

Multi-head attention

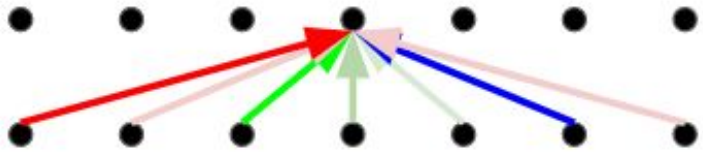
- Multiple attention layers (heads) that run in parallel
- Each head use different weights
- Each head can learn different relationship



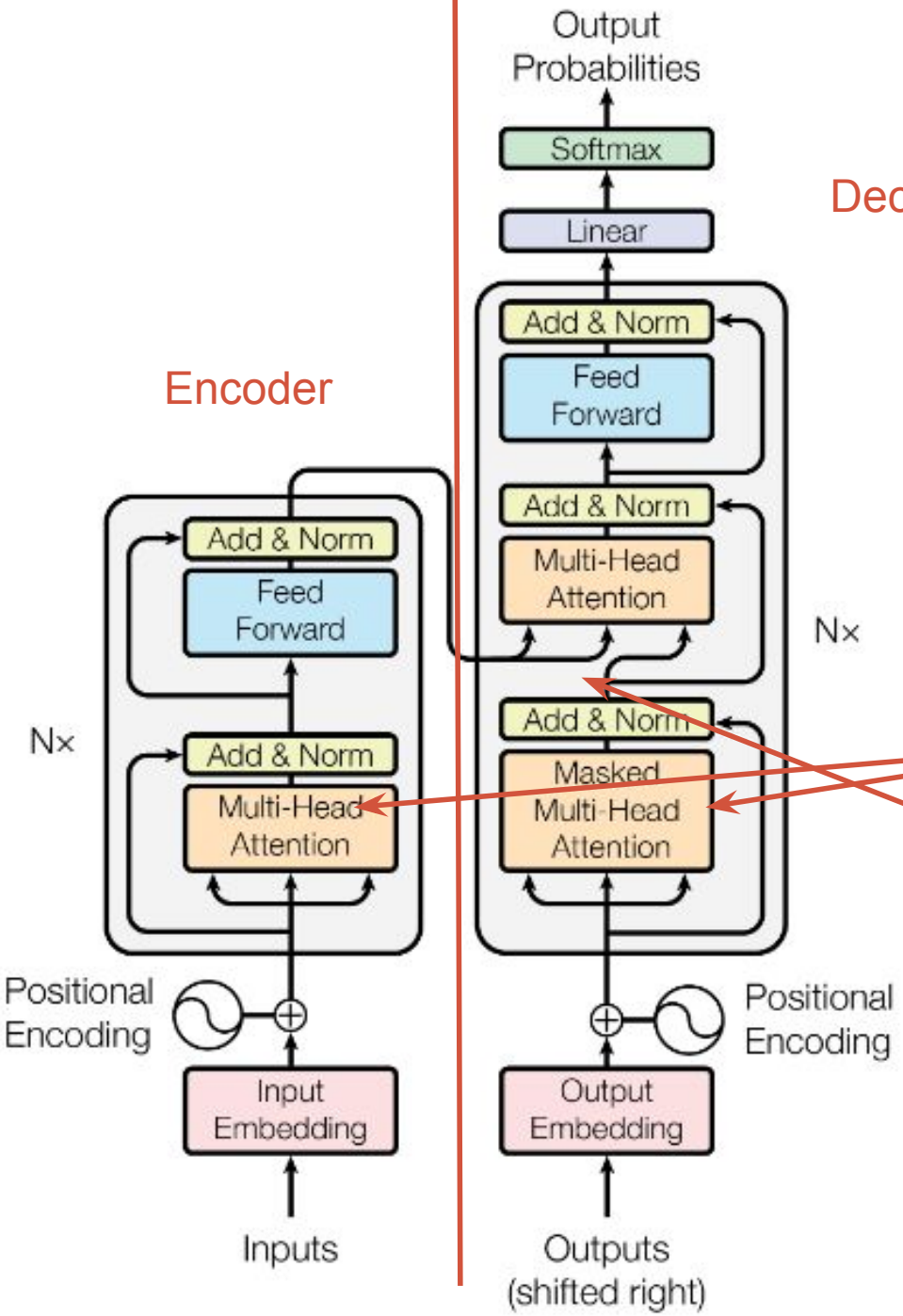
Convolution

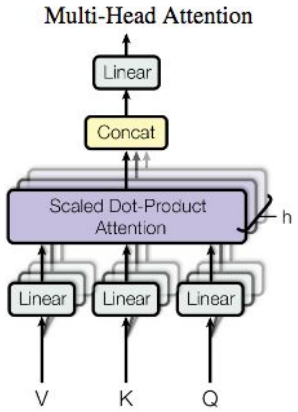


Multi-Head Attention



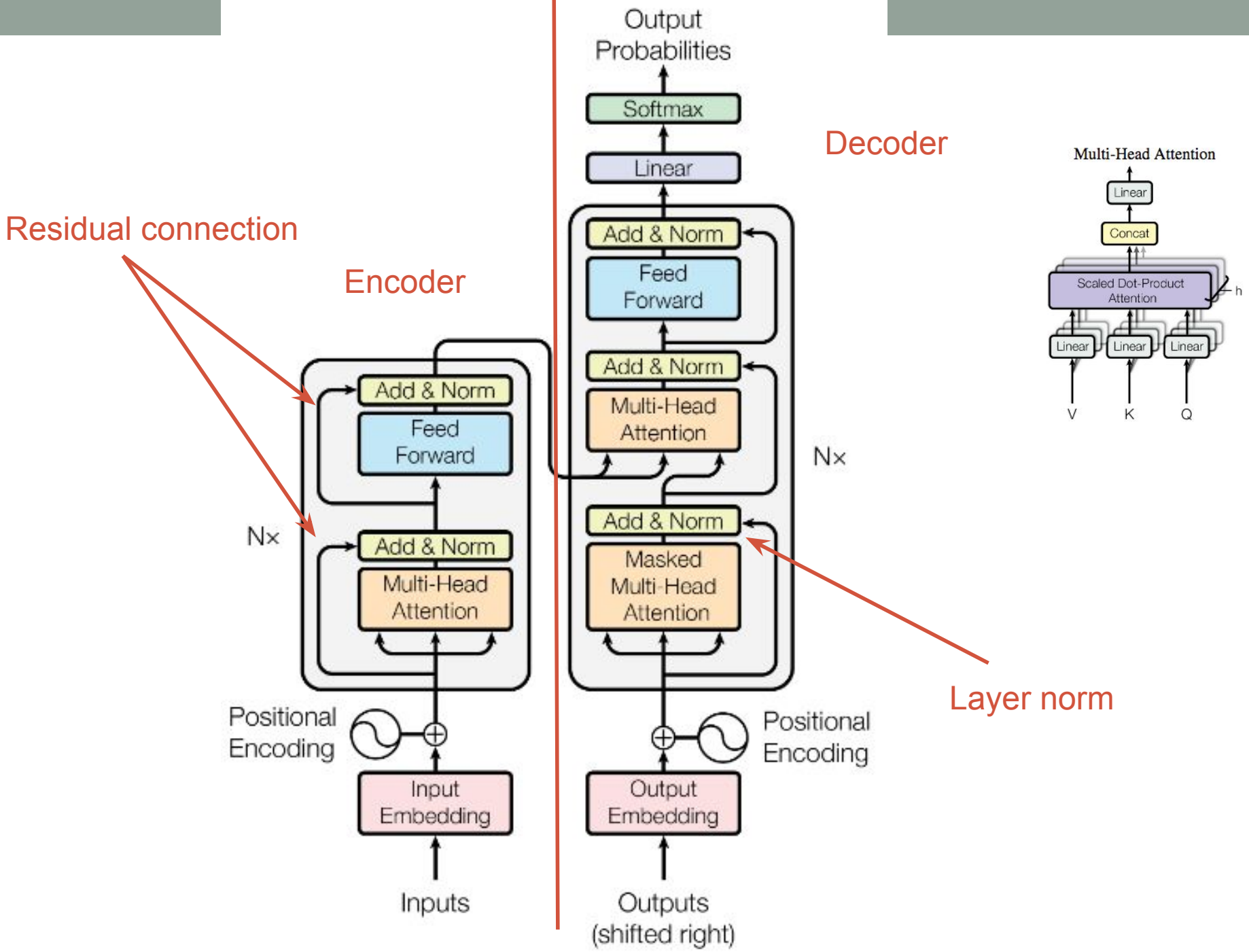
Multi-head visualization





Self attention

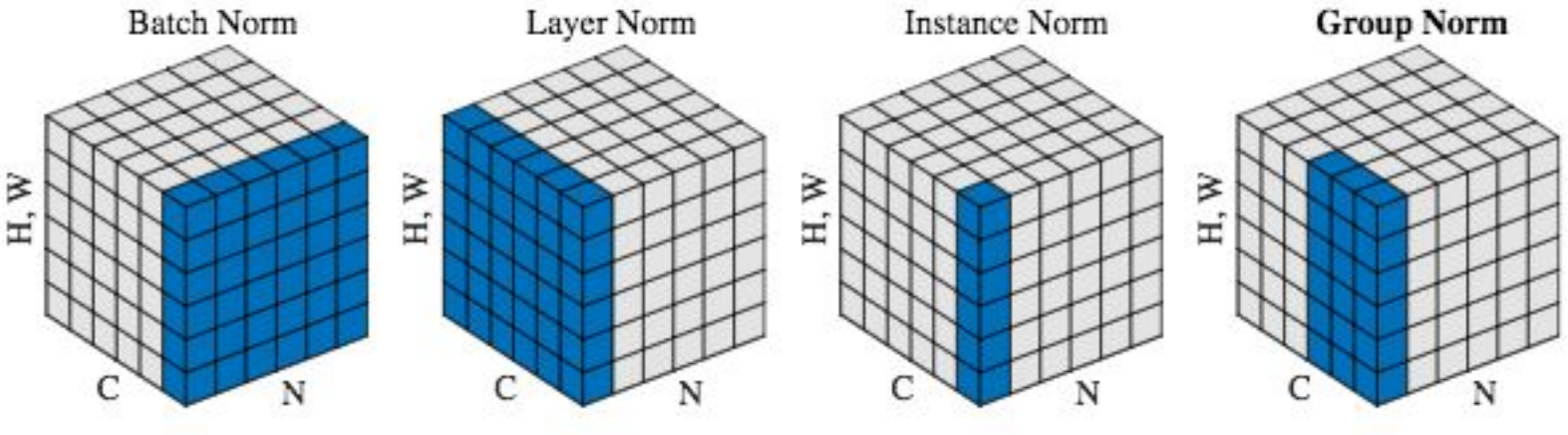
Encoder gives Value and Key
The decoder gives Query


Residual connection
Encoder
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Layer norm
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

Layer norm

- Normalize the mean and SD
- Batch norm vs layer norm vs Instance norm vs group norm

Group is used to distributed models into multiple GPUs

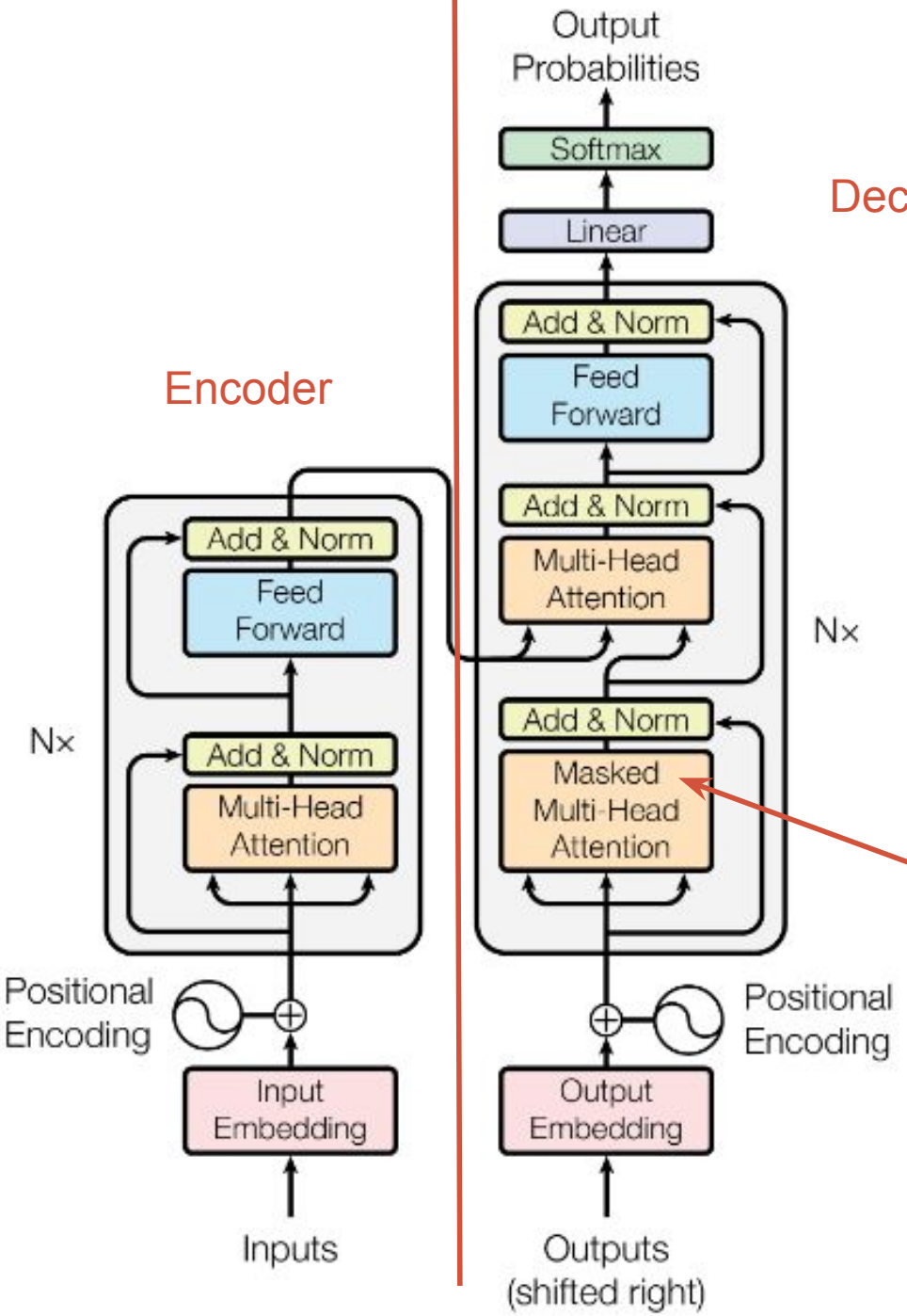


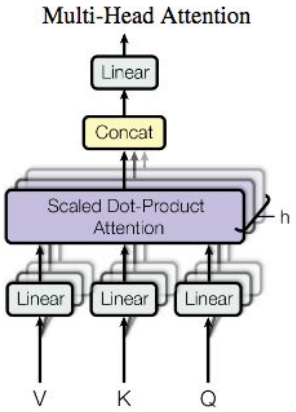
N – example in mini batch
C – Channel output
H,W – spatial coordinates (x,y)

Box is output tensor from CNN

BN and GN are usually best, GN is better when batch size is small (Vision task)

https://arxiv.org/abs/1803.08494





Prevent looking ahead (cheating)

MT results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Can use for other tasks, like ASR, parsing, etc.

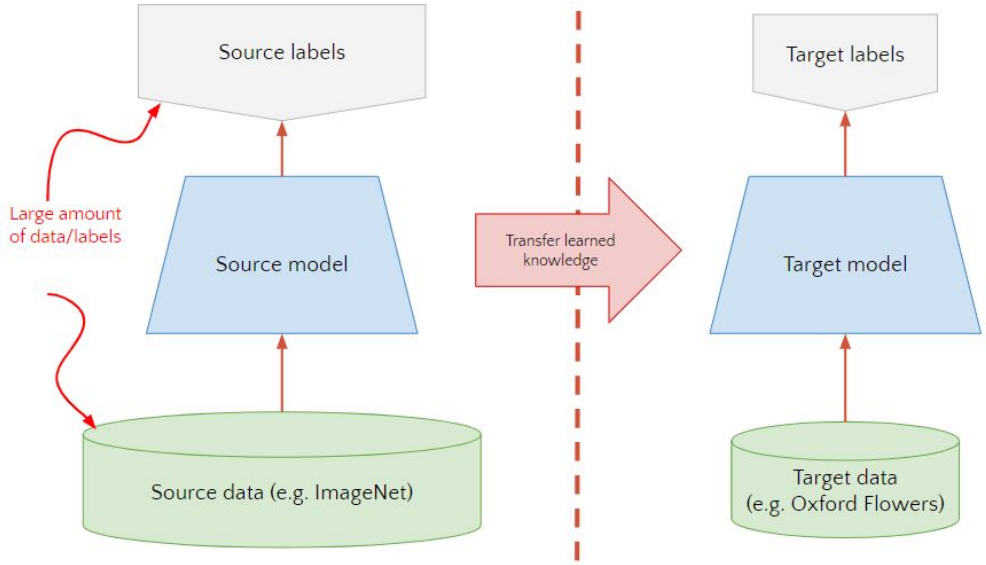
Misc topics

- Encoder-Decoder
- Attention modeling
- Transfer Learning, Multi-task models, Multi-models
- Object detection

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

Transfer learning (basics)

- We know networks captures good representations
- Can we use it for other tasks?
- Use trained networks to initialize a new network for a different task.
- Re-train the network using SGD on new data.



For CV tasks, we call the pre-trained network backbones

Transfer learning idea

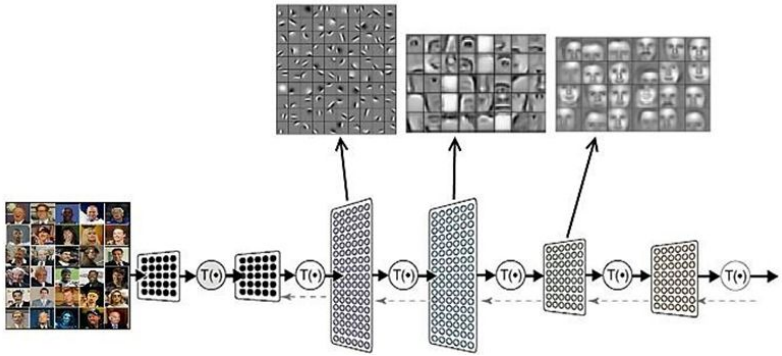
Instead of training a deep network from scratch for your task, you can

- Take a network trained on **a different domain** for a **different source task**
- **Adapt (fine-tune)** it for your domain and your **target task**

This lecture will talk about how to do this.

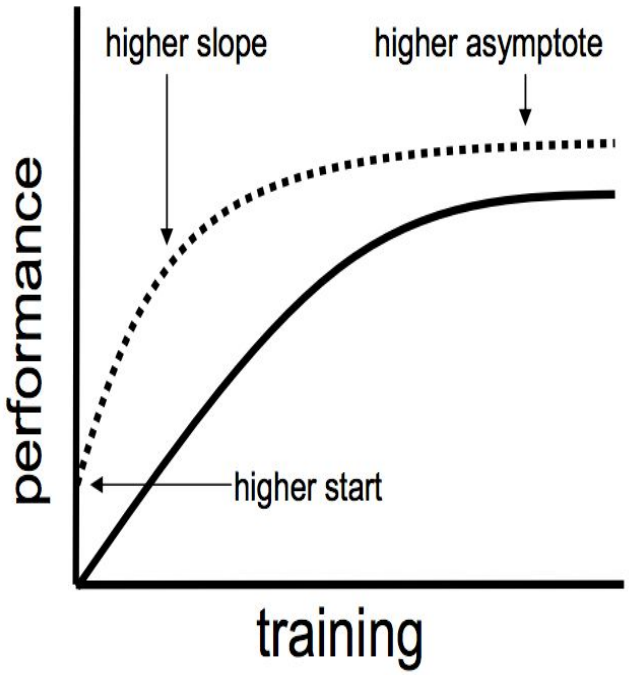
Variations:

- Different domain, same task
- Different domain, different task



Benefits to transfer learning

1. Higher start
2. Higher slope (converge faster)
3. Higher asymptote (if small data)

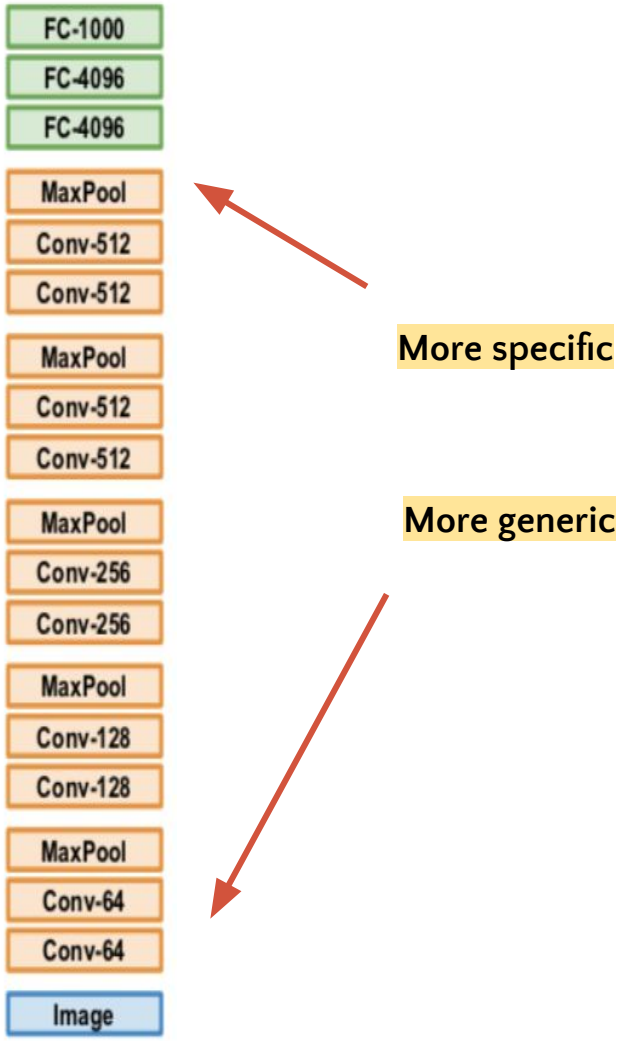


Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.

Layers

Lower layers: more general

Higher layers: more specific

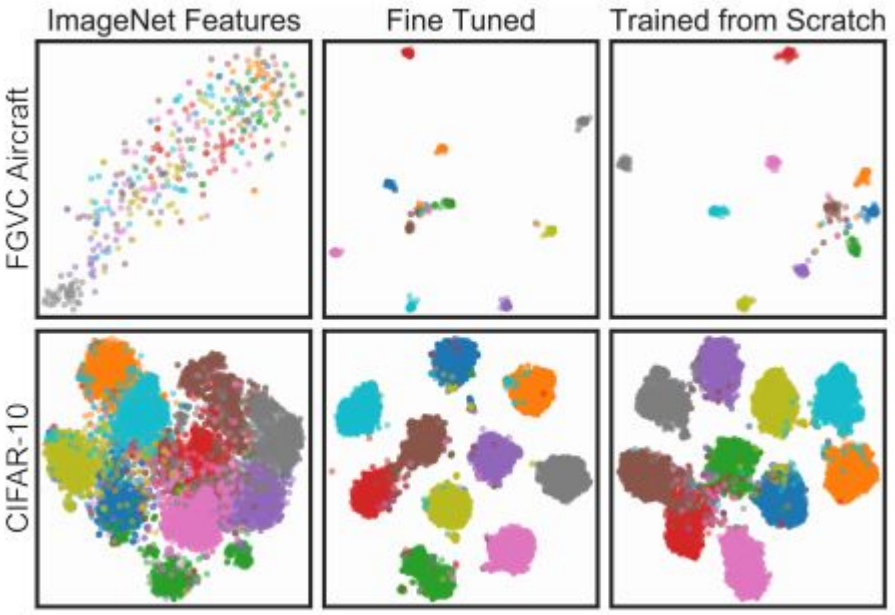


Domains

Similar domain can transfer easier

Fine-tuning (adaptation) is crucial

Aircraft images
Different from ImageNet



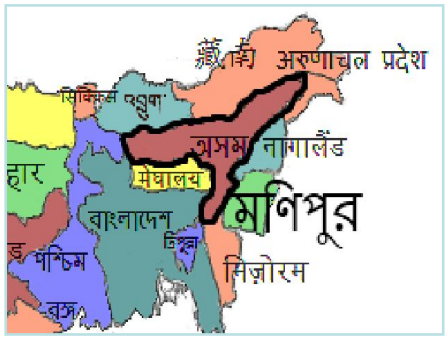
Natural images
Similar to ImageNet

Kornblith et al., Do Better ImageNet Models Transfer Better?, arxiv 2018 https://arxiv.org/abs/1805.08974

Transfer learning in ASR

Assamese and Bengali language

Bengali



Assamese

Bengali	WER
No pre-training (10 hr Bengali)	71.8%
No pre-training (60 hr Bengali)	64.5%
Transfer learning (10 hr Assamese -> 10 hr Bengali)	66.0%
Transfer learning (60 hr Assamese -> 10 hr Bengali)	64.6%
+ Adaptation	63.7%
Transfer learning (60 hr Bengali -> 10 hr Bengali)	61.6%

Closer domain is better

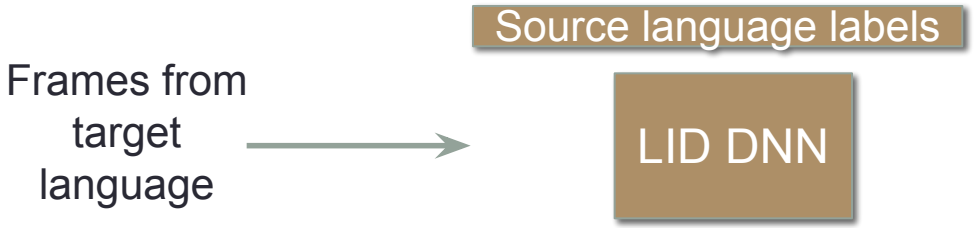
- Pre-train on 60 hours of data, and adapt on languages with 10 hours of data
- Numbers in blue is for 10->10 hours (baseline)

		Pretrain on			
		Bengali	Assamese	Lao	Turkish
Adapt to	Bengali	66.0	63.8	65.1	64.2
	Assamese	61.2	65.2	62.9	62.1
	Lao	59.8	60.1	62.3	60.0
	Turkish	61.8	63.1	63.3	63.9

- Transfer learning always improves performance.
- Similar languages perform better on transfer learning
- Can we identify this automatically?

Language Identification for data selection

- Train a classifier (LID) on source languages to predict the language given input frames
- Compute posteriors of the target language data using that classifier
- The best language for the target language should have the highest average posterior score



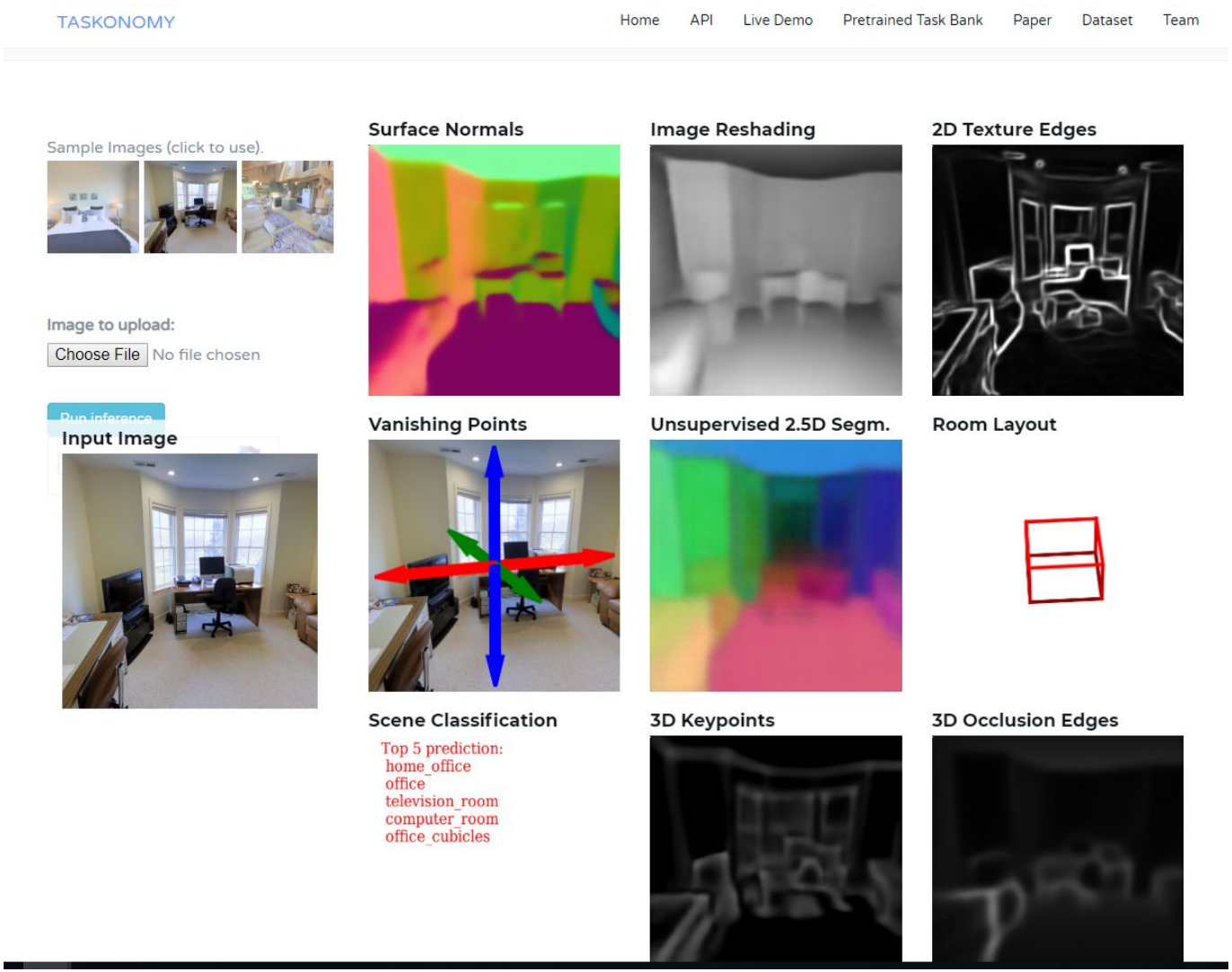
Predicting the best language

		Pretrain on			
		Bengali	Assamese	Lao	Turkish
Adapt to	Bengali	66.0	63.8	65.1	64.2
	Assamese	61.2	65.2	62.9	62.1
	Lao	59.8	60.1	62.3	60.0
	Turkish	61.8	63.1	63.3	63.9

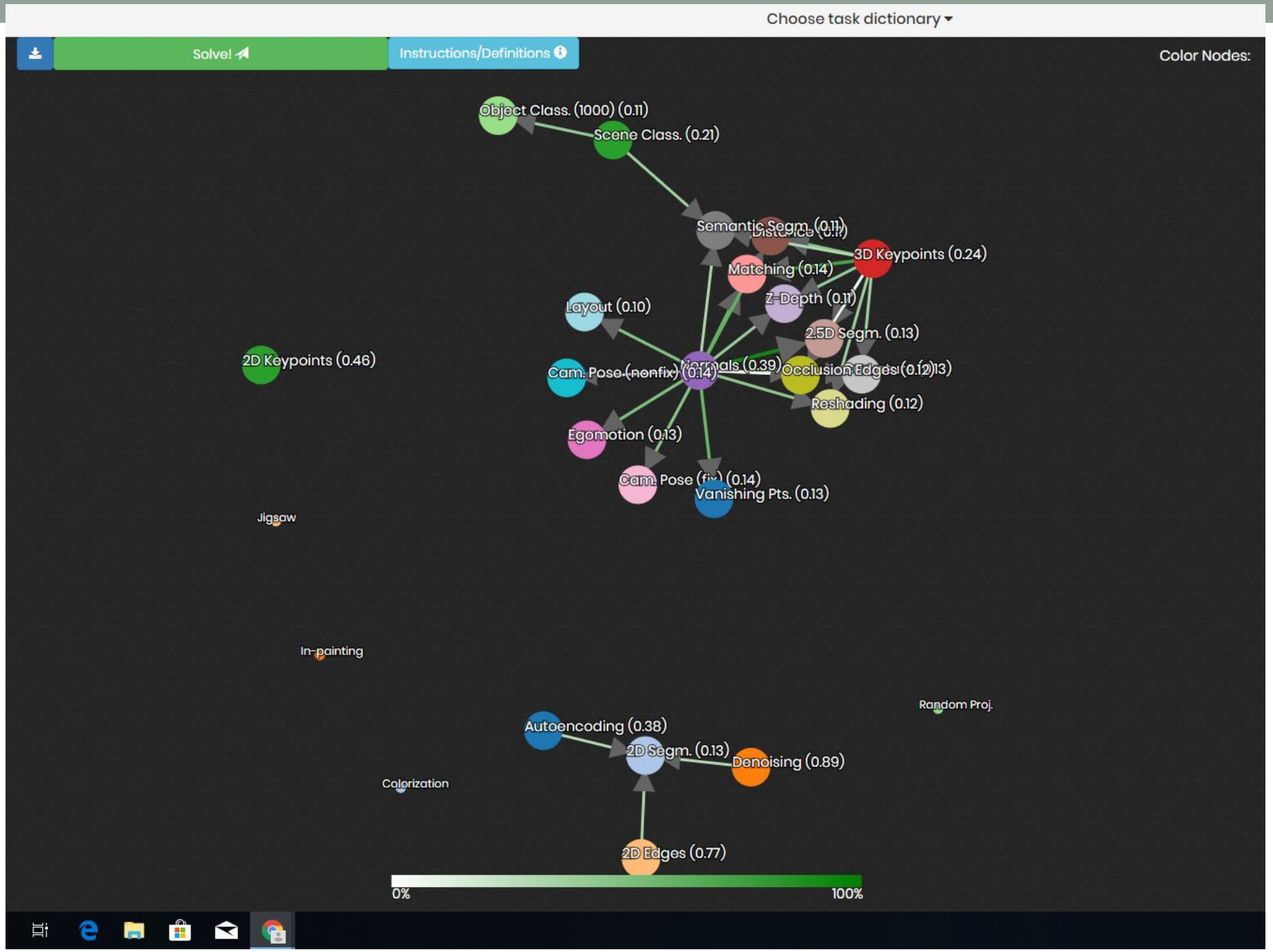
		Language prediction score			
		Bengali	Assamese	Lao	Turkish
Input frames	Bengali	0.57	0.21	0.09	0.13
	Assamese	0.21	0.57	0.11	0.11
	Lao	0.08	0.11	0.71	0.10
	Turkish	0.13	0.12	0.10	0.65

The LID scores correspond to the best language to use most of the time.

Which task to transfer?



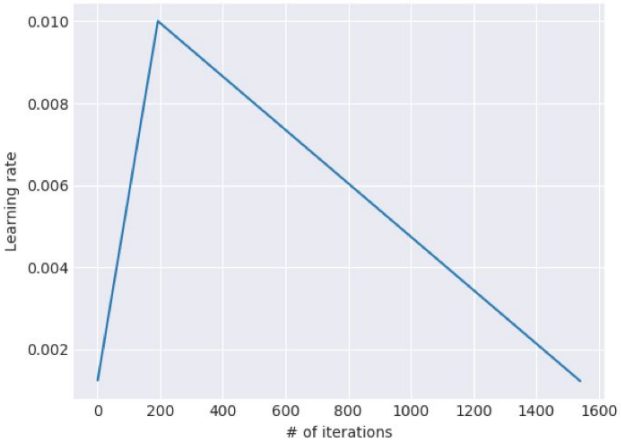
http://taskonomy.stanford.edu/tasks/


Choose task dictionary
Solve!
Instructions/Definitions
Color Nodes:
Object Class. (1000) (0.11)
Scene Class. (0.21)
Semantic Segm. (0.11)
3D Keypoints (0.24)
Matching (0.14)
Z-Depth (0.11)
Layout (0.10)
2.5D Segm. (0.13)
2D Keypoints (0.46)
Cam. Pose (nonfix) (0.14)
Occlusion Edges (0.12)
Reshading (0.12)
Egomotion (0.13)
Vanishing Pts. (0.13)
Jigsaw
In-painting
Random Proj.
Autoencoding (0.38)
2D Segm. (0.13)
Denoising (0.89)
Colorization
2D Edges (0.77)
0%
100%

Adaptation tricks

Triangle learning rate

At the start some of the model is randomly initialized. Cannot trust the gradient



Discriminative fine-tuning

Instead of using the same learning rate for all layers of the model, discriminative fine-tuning allows us to tune each layer with different learning rates.

- Layers that are closer to inputs → large learning rate
- Layers that are closer to outputs → small learning rate

Advance adaptation ideas

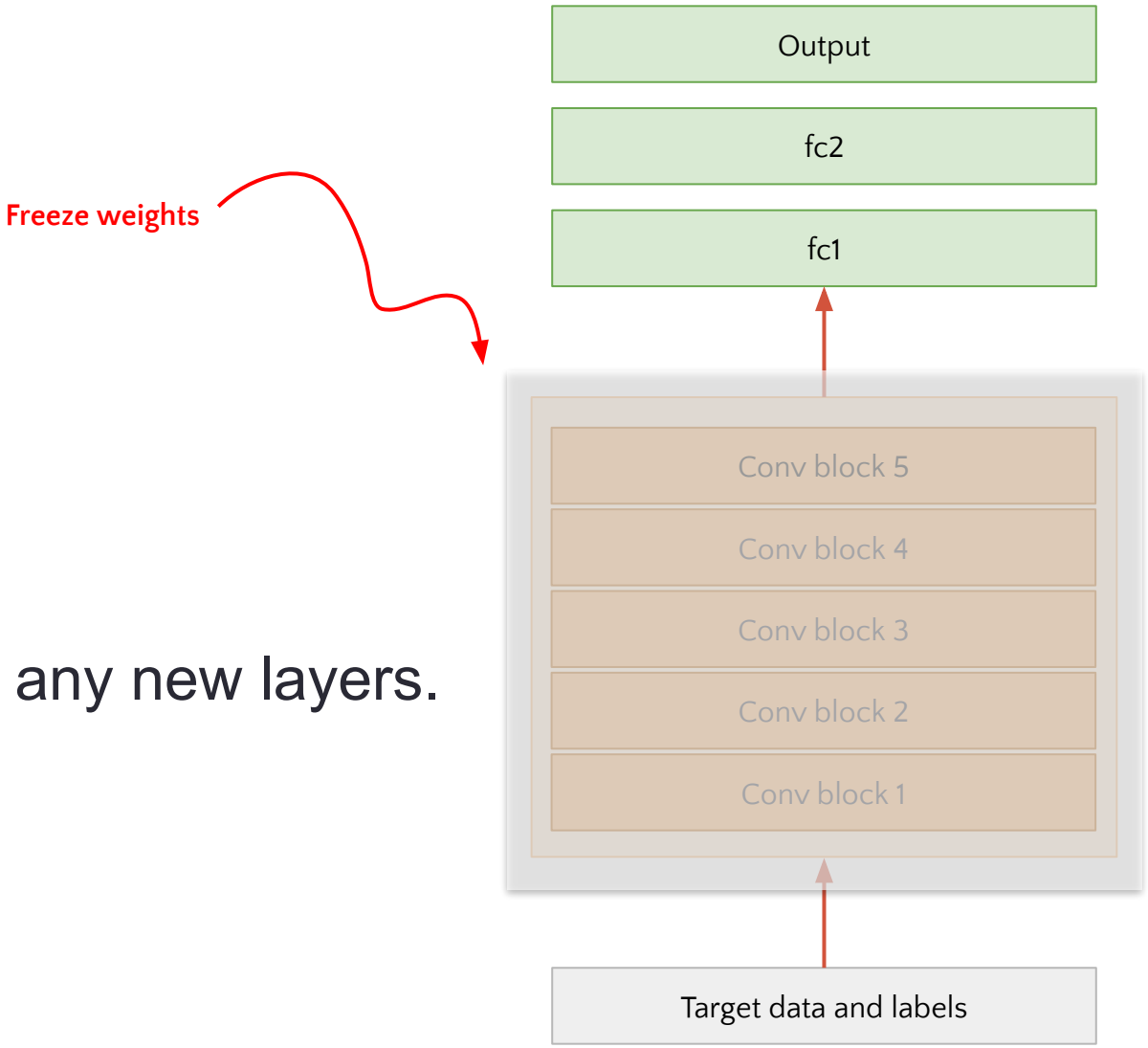
- Chain-taw
  - Freeze the old layers, train the new weights for a bit

1. fine-tunes any new layers.
2. fine-tunes each layer individually of base model starting from the first to the last.
3. the entire model is trained with all layers.

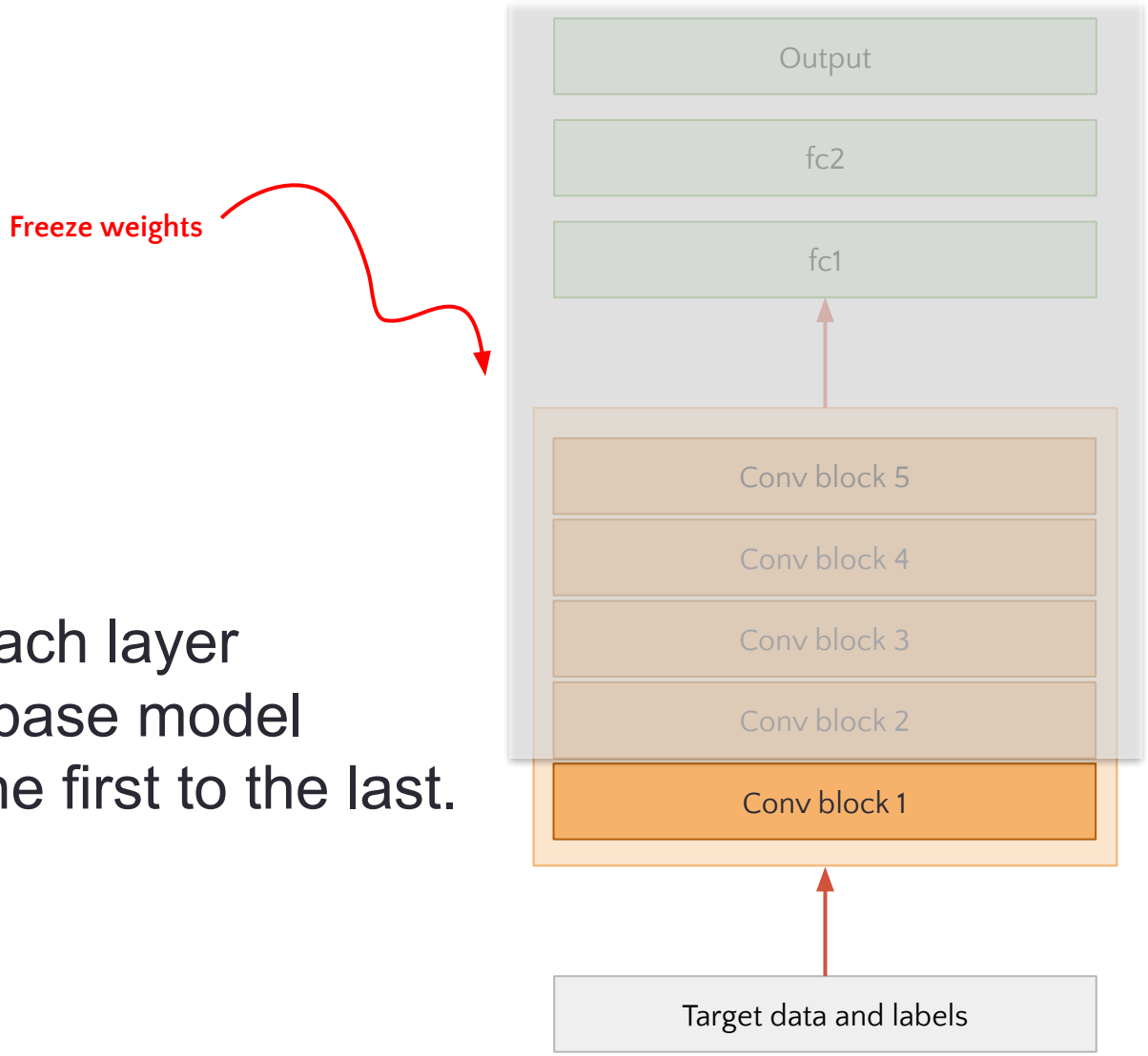
Felbo et al., **Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,** arxiv 2017, https://arxiv.org/pdf/1708.00524.pdf

Chain-taw

1. fine-tunes any new layers.

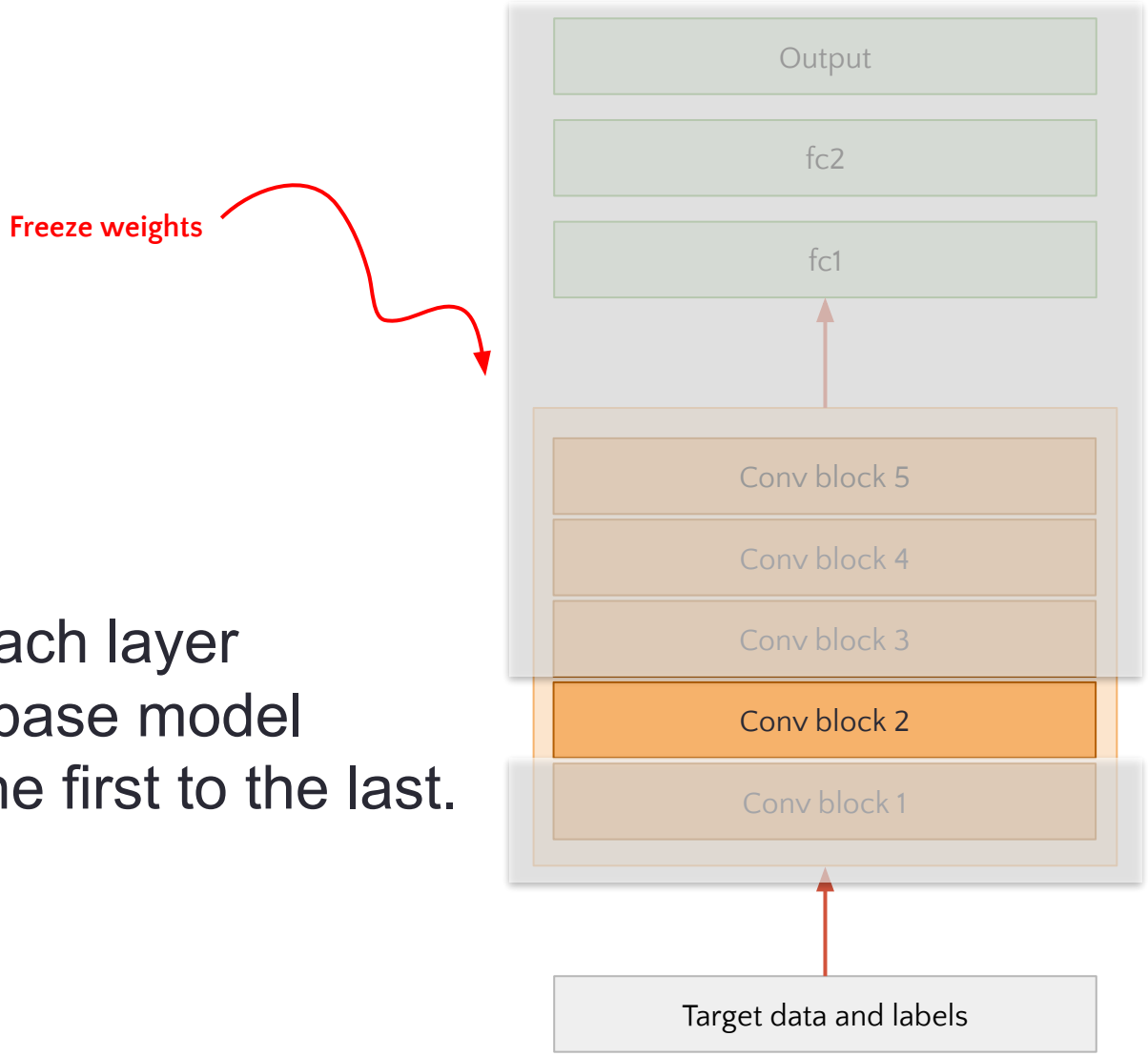


Chain-taw



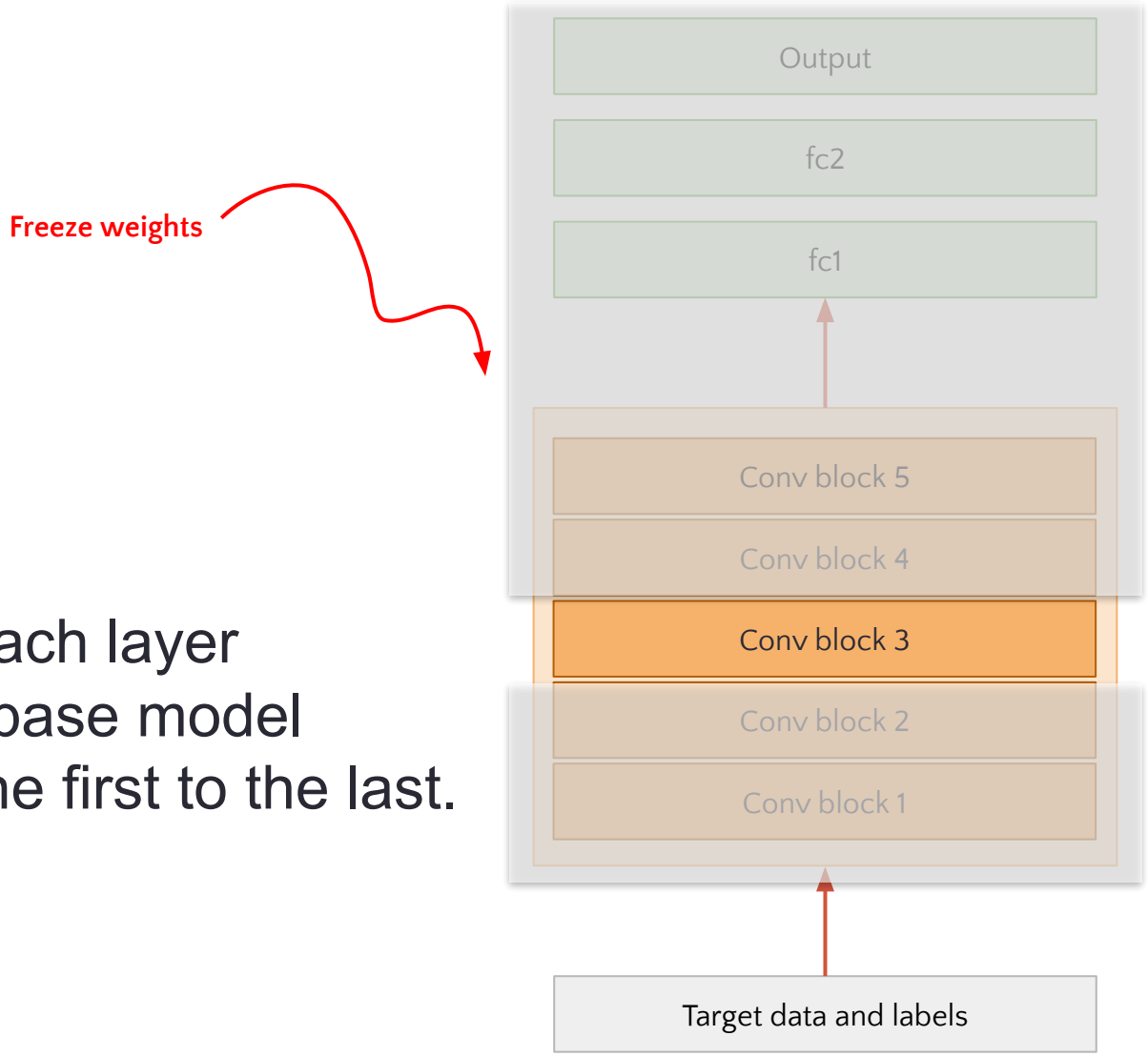
2. fine-tunes each layer individually of base model starting from the first to the last.

Chain-taw



2. fine-tunes each layer individually of base model starting from the first to the last.

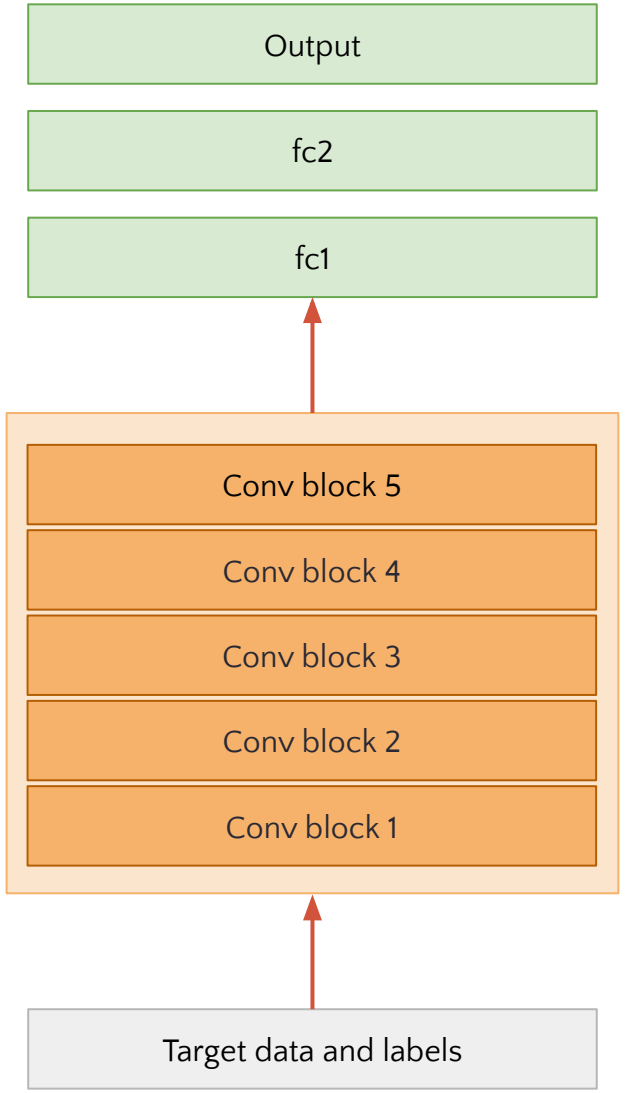
Chain-taw



2. fine-tunes each layer individually of base model starting from the first to the last.

Chain-taw

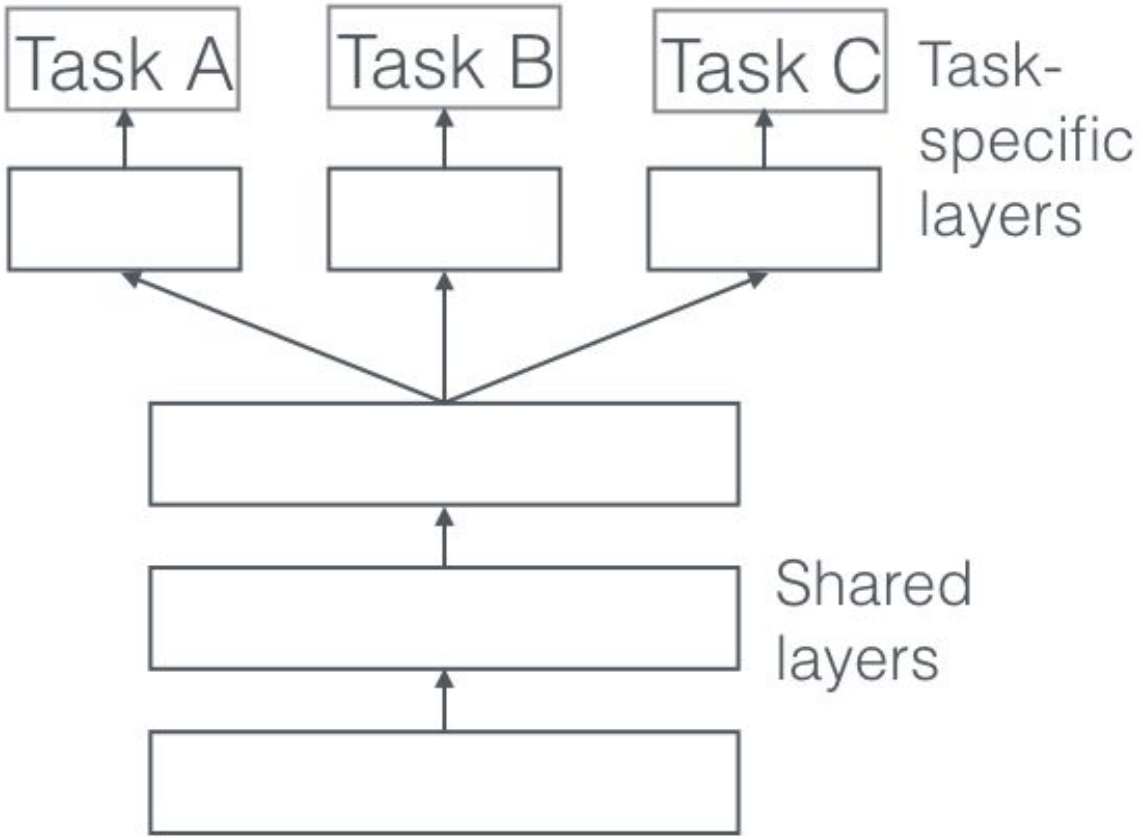
3. the entire model is trained with all layers.



Multi-task training (Joint models)

- Training a neural network on multiple tasks at the same time can help the network distinguish between noise and important information
- Helps partition the output space
- Example: Input speech, output text vs Input speech, output text and language.

Multi-task learning (simplest form)



Total Loss = W0 Loss A + W1 Loss B + W2 Loss C

http://ruder.io/multi-task/

Why multi-task?

Gives additional information for the network to learn the structure of the data

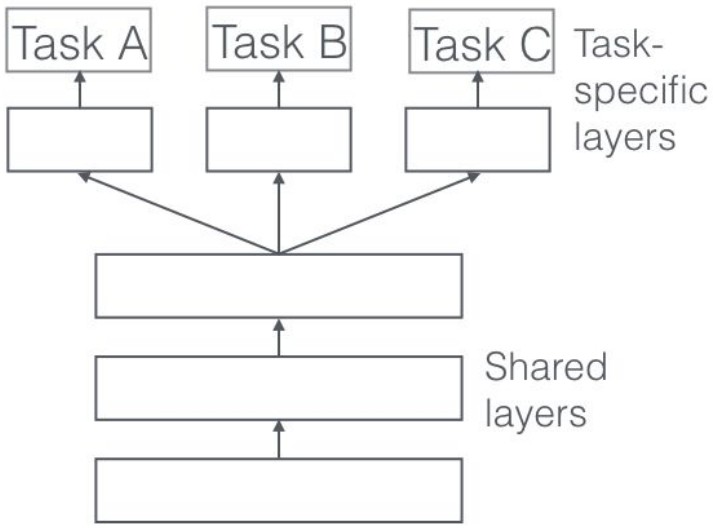
Training images




Test image



Multi-task learning



Multiple setups:

-Complete information

$(x_1, y_{1A}, y_{1B}, y_{1C})$ $(x_2, y_{2A}, y_{2B}, y_{2C})$

Straightforward to implement

-Incomplete information

(x_1, y_{1A}, y_{1B}) (x_2, y_{2C})

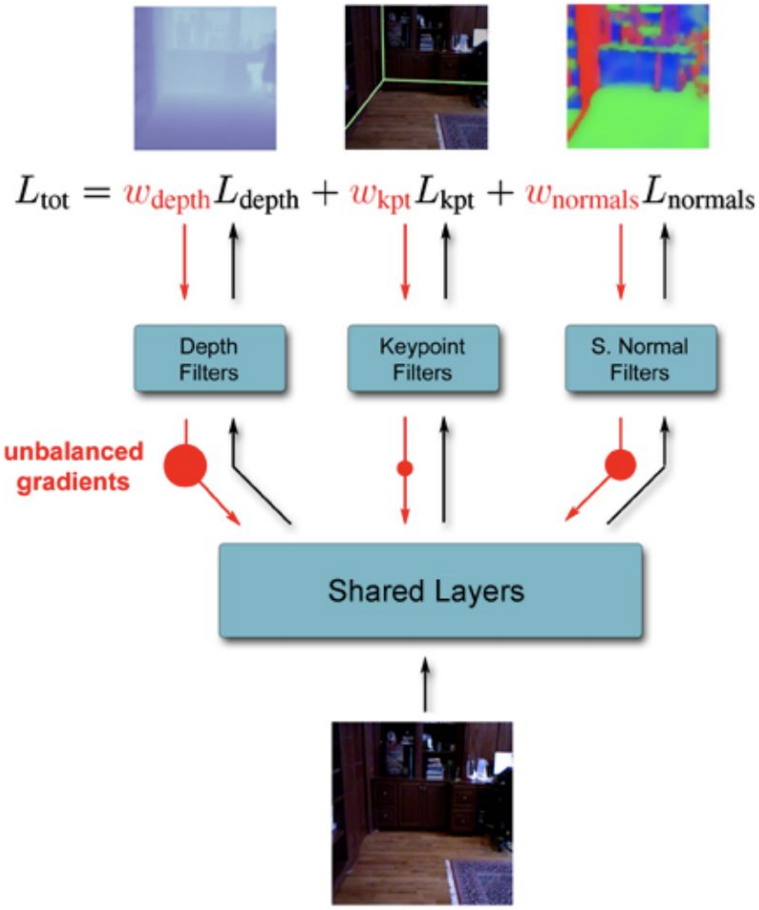
More realistic, can handle multiple datasets

Need to shuffle between mini-batches

Mask unused task heads

Joint loss

- The loss function of multi-task is usually a linear combination of each head
- Needs to adjust the weights so that no single head dominates the gradients
- **Very important for performance**
- How to select the weights?



https://arxiv.org/abs/1711.02257

Weights selection

- Normalizing the loss (easy)
  - Try to make the Loss of each head be of similar magnitude
- Normalizing the gradients (slightly harder)
  - Try to make the gradients from each head be of similar magnitude
- Grid search (lazy programmer)
  - N heads mean N parameters search

What if the weights should be change over time?

Other ideas

Normalize by how much the task have learned

GradNorm (https://arxiv.org/abs/1711.02257)

Loss of the head decreases a lot → the task learn too much

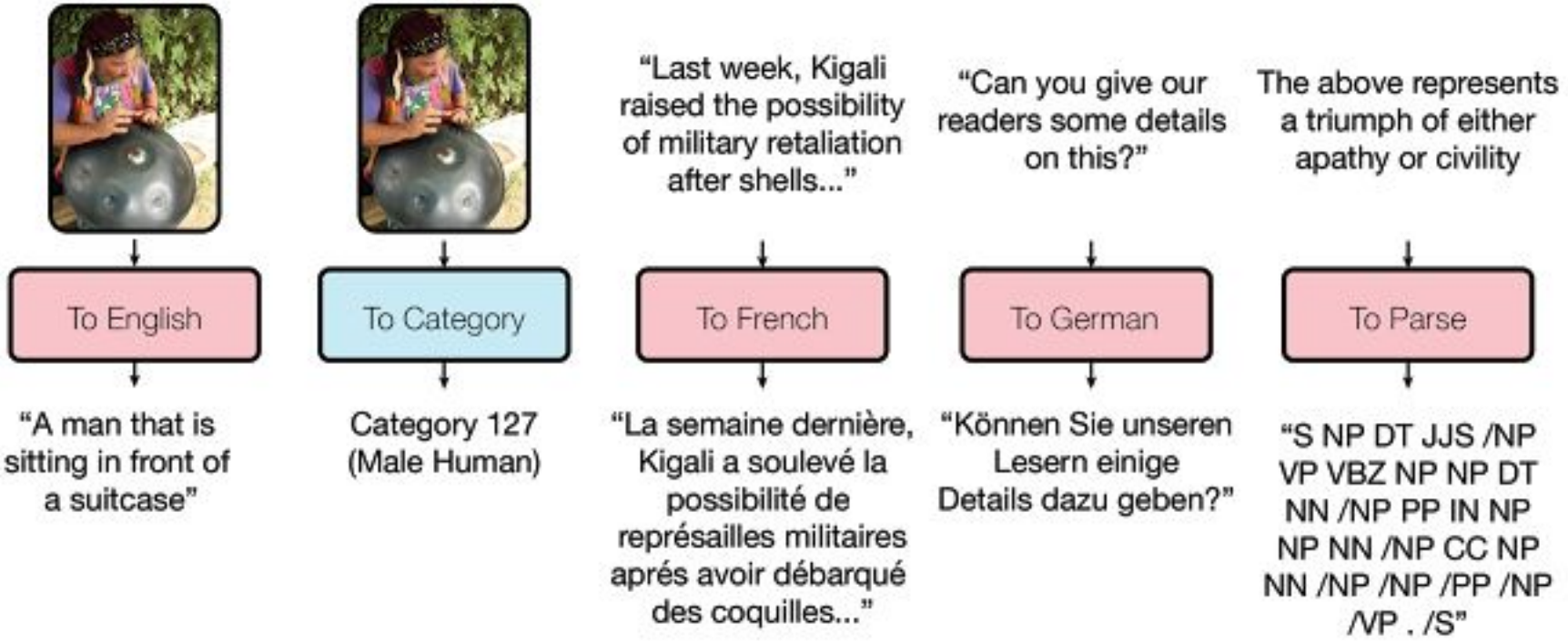
Task prioritization

(http://openaccess.thecvf.com/content_ECCV_2018/papers/Michelle_Guo_Focus_on_the_ECCV_2018_paper.pdf)

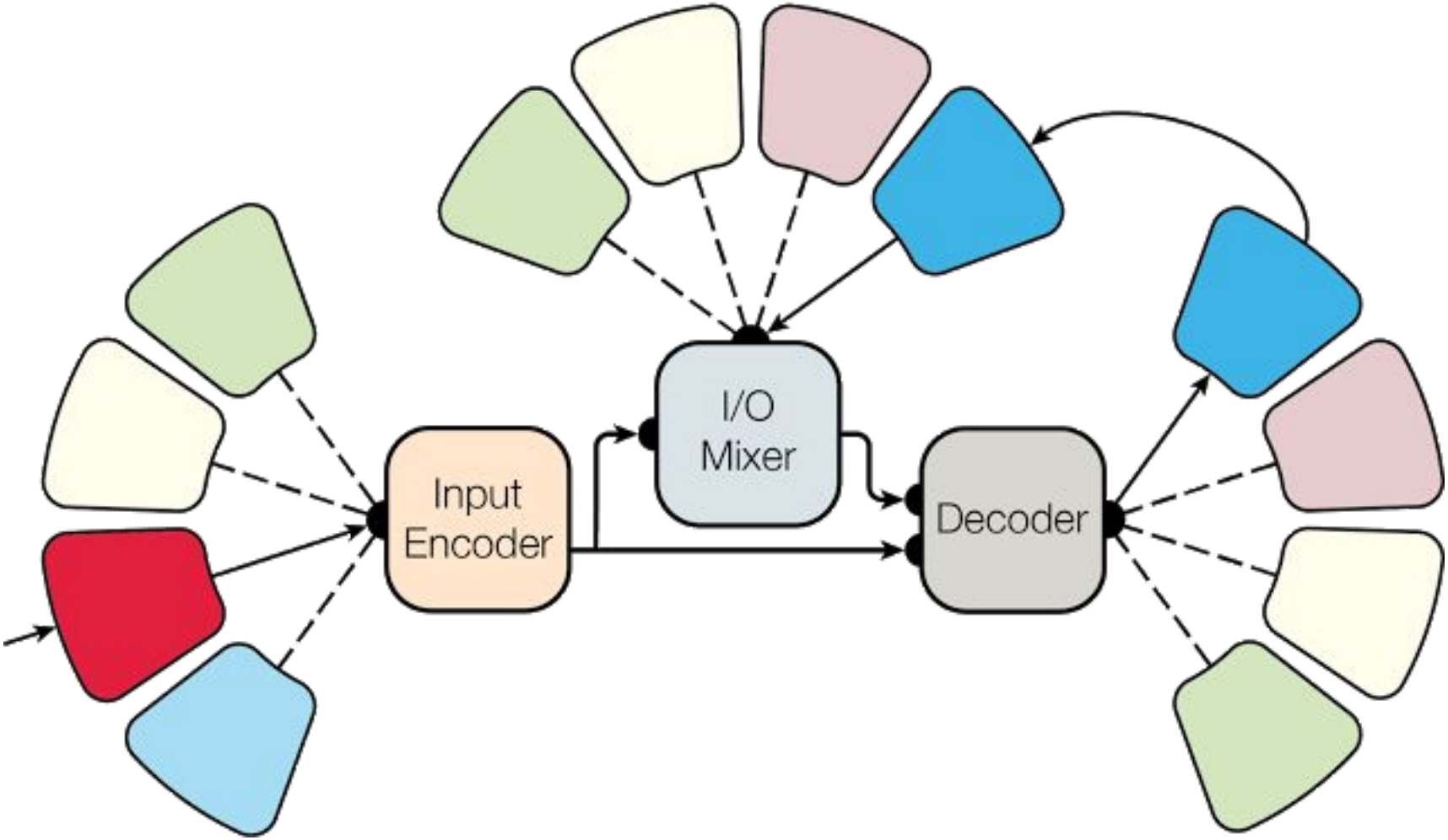
Rather than loss, look at how good the task is (task-KPI)

Multimodels

- A single model that can handle different kinds of inputs
- Joint representation of meaning improves performance



https://research.googleblog.com/2017/06/multimodel-multi-task-machine-learning.html


I/O
Mixer
Input
Encoder
Decoder

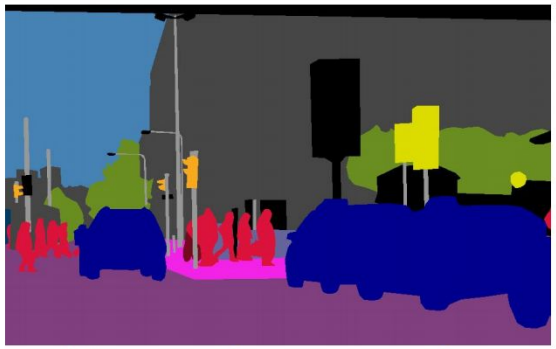
Misc topics

- Encoder-Decoder
- Attention modeling
- Transfer Learning, Multi-task models, Multi-models
- Object detection

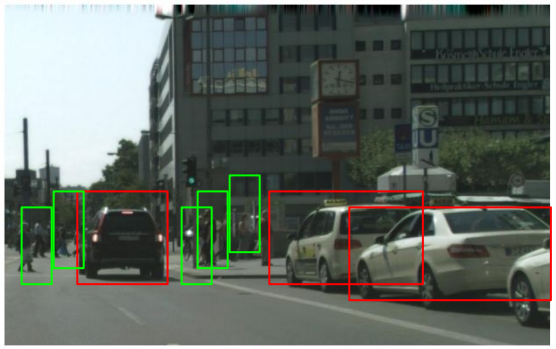
- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

Image tasks

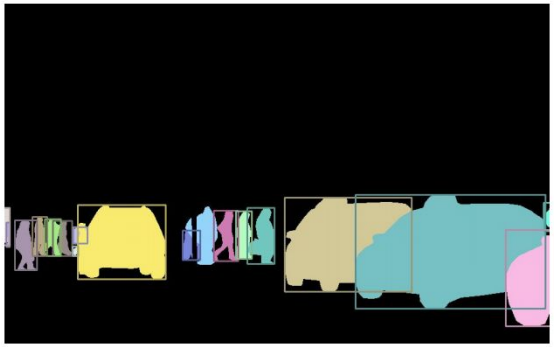
(a) image



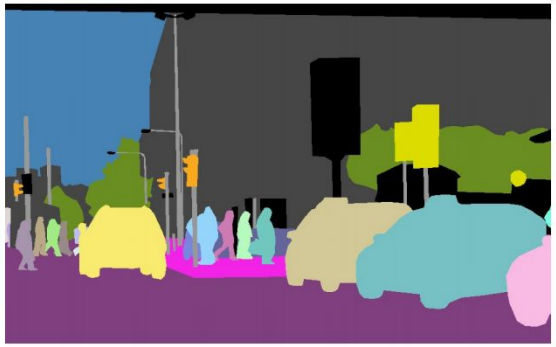
(b) semantic segmentation



object detection



(c) instance segmentation

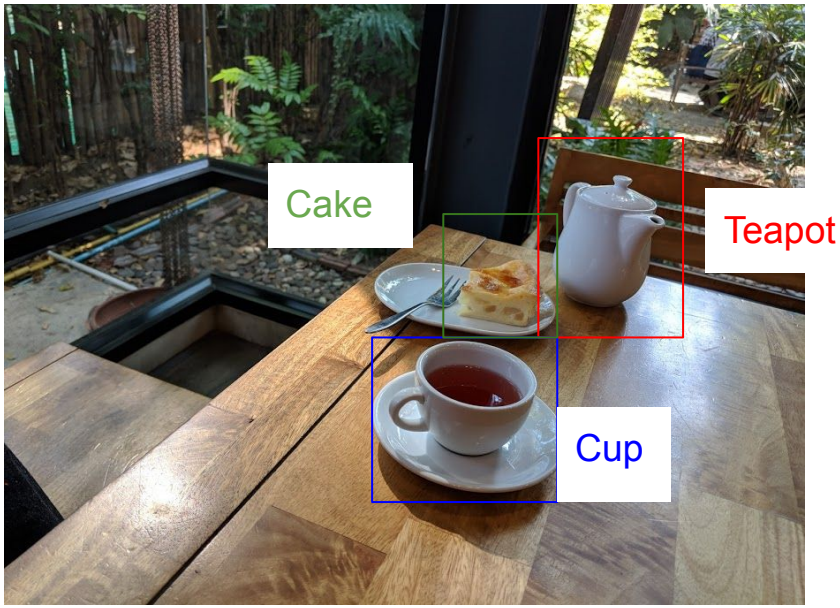


(d) panoptic segmentation

https://towardsdatascience.com/panoptic-segmentation-with-upsnet-12ecd871b2a3

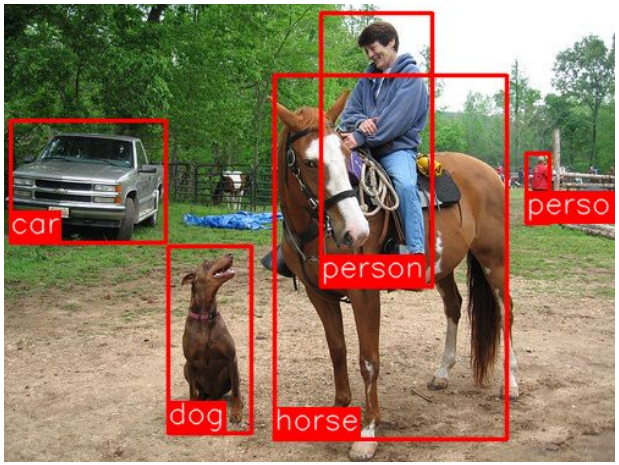
Object detection

Locate every object in an image and identify them



Object detection

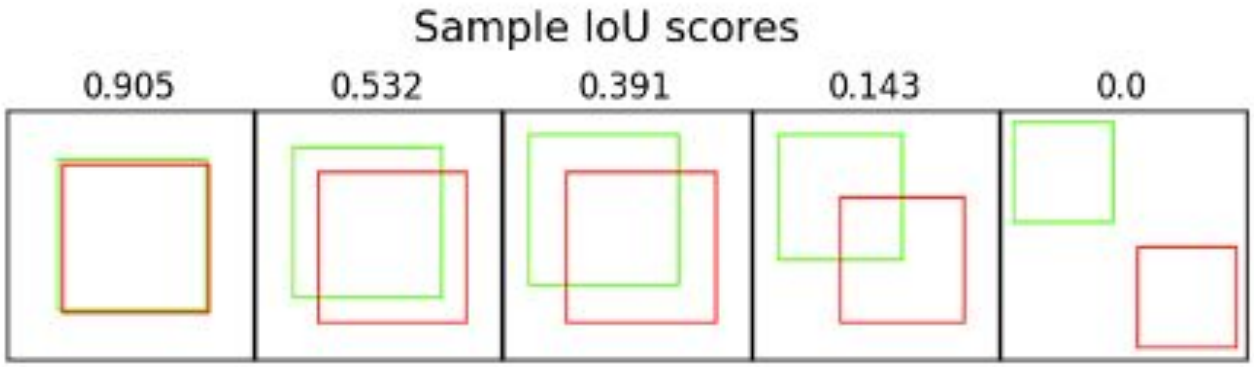
- Identify all objects in an image
- Two sub-task
  - Determine what objects
  - Determine where objects are



https://arxiv.org/abs/1311.2524.

IoU (Intersection over Union)

What IoU score should be considered a detection?

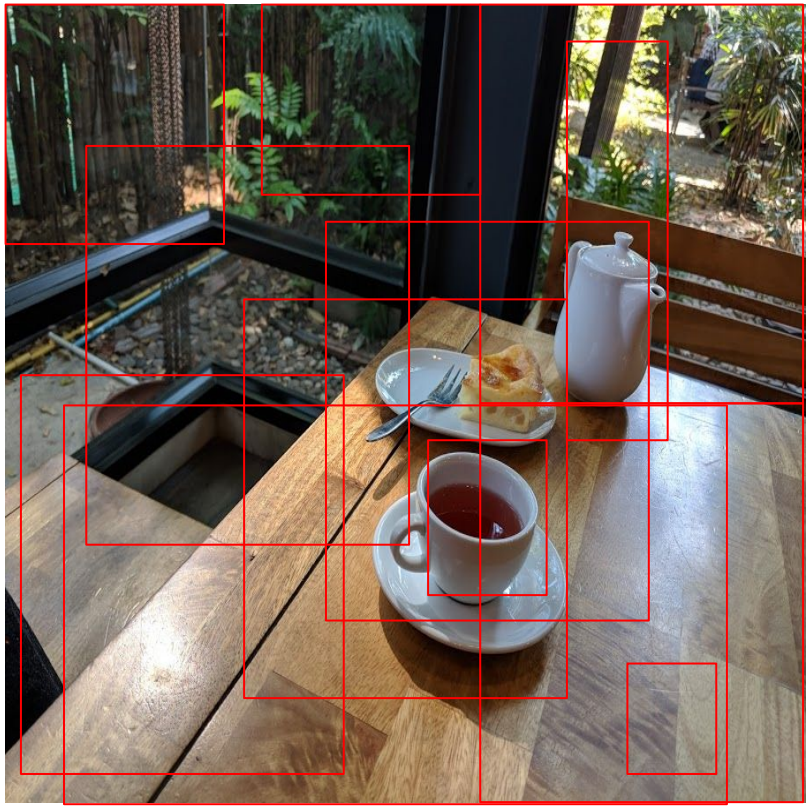


http://ronny.rest/tutorials/module/localization_001/iou/

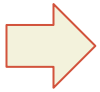
Simplest solution: sliding window

Slide a object classifier over every possible region and scale

Expensive. Can we do better?



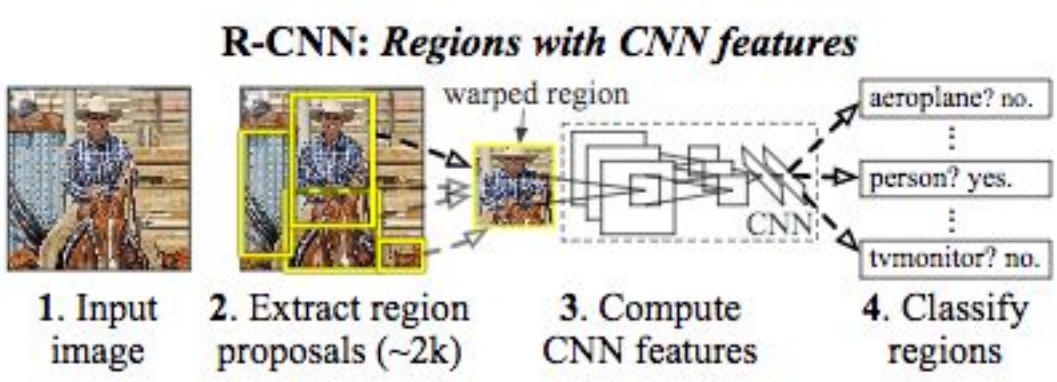
Object classifier (backbone)



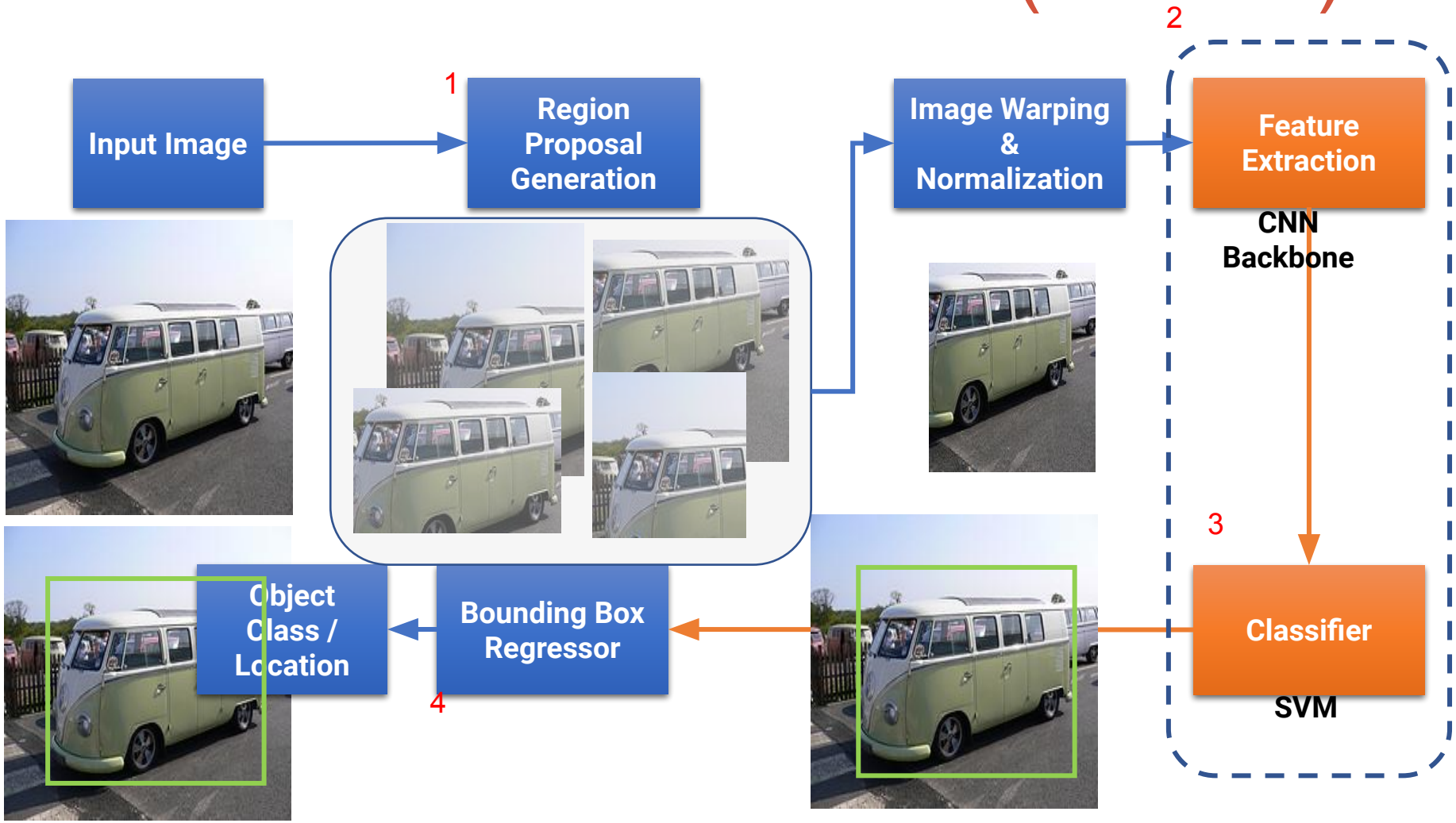
Candidate lists with probabilities

Object Detection Comprehensive Review

1. Proposal-based detector
   - R-CNN
   - Fast-RCNN
   - Faster-RCNN
   - Mask RCNN
2. Single stage detector
   - YOLO
   - SSD



Regions with Convolutional Neural Network Feature (R-CNN)



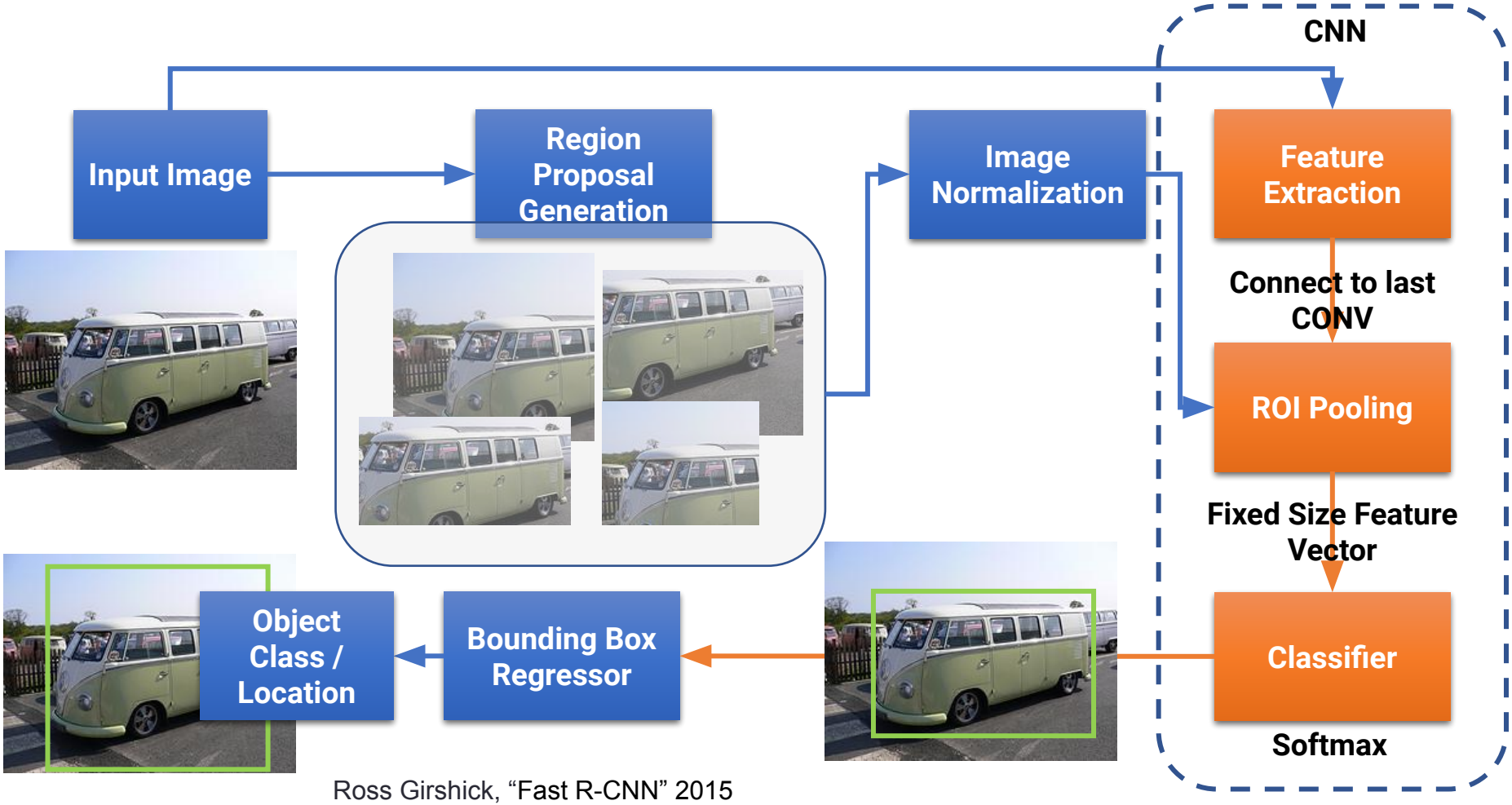
Ross Girshick, “Rich feature hierarchies for accurate object detection and semantic segmentation” 2013.

Regions with Convolutional Neural Network Feature (R-CNN)

- Region proposal generation (selective search) is computationally expensive.
- No computation sharing between regions.
- In the training process, R-CNN suffers from the following complex multi-stages training.
    - The backbone needs to be adapted to the new task (detection and regression)
    - The SVM classifier
    - The bounding box regressor

Fast-RCNN

- Computation sharing for each region proposal by computing the feature extraction once then extract features for each region.
- A novel Region Of Interest Pooling (ROI Pooling) layer.



Ross Girshick, “Fast R-CNN” 2015

Fast-RCNN

Feature Extraction

Take features directly from CONV layer

Corresponding location respect to region proposal location

Last CONV output feature maps

Selective Search

ROI pooling

1
2
3
5
78
90
15
..

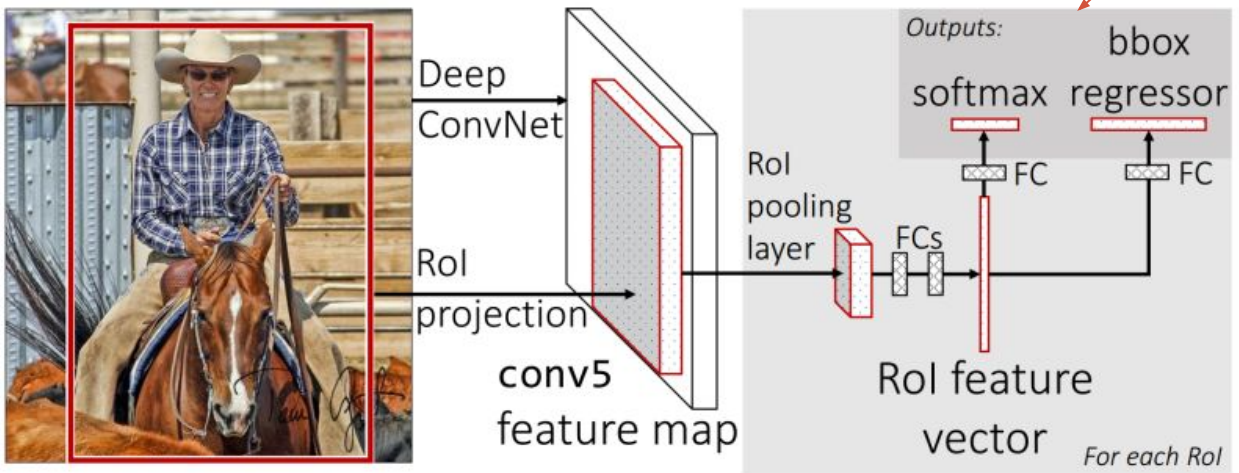
Fixed size feature vector

Divided into fixed partition and take max value

Fast-RCNN

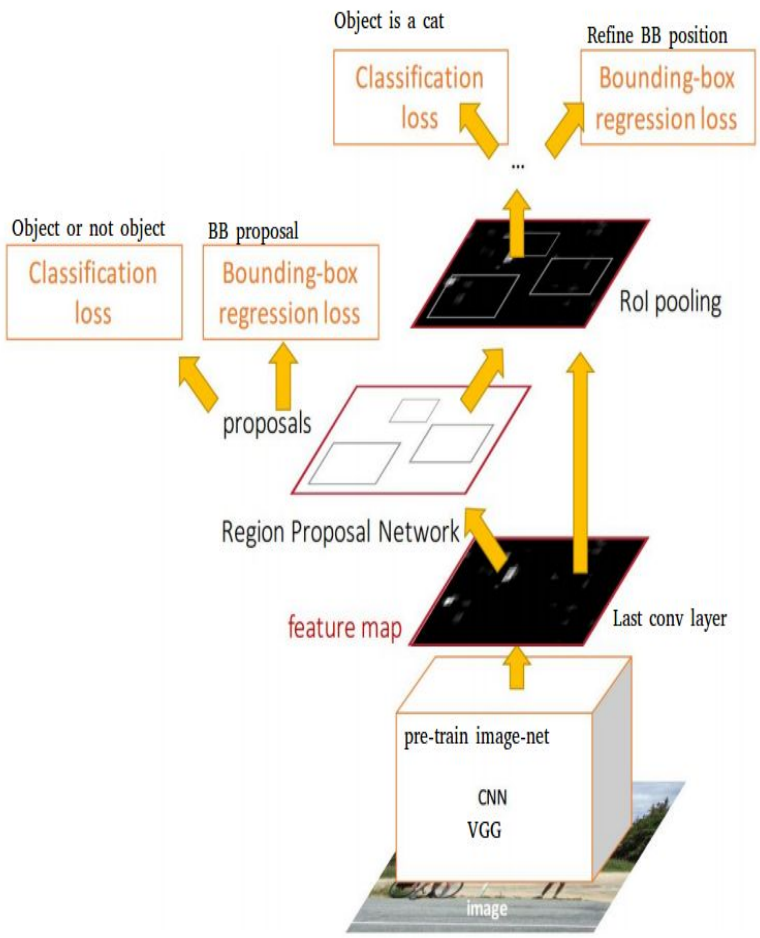
- Fast R-CNN still relay on external object proposal algorithm (Selective Search)

Selective Search



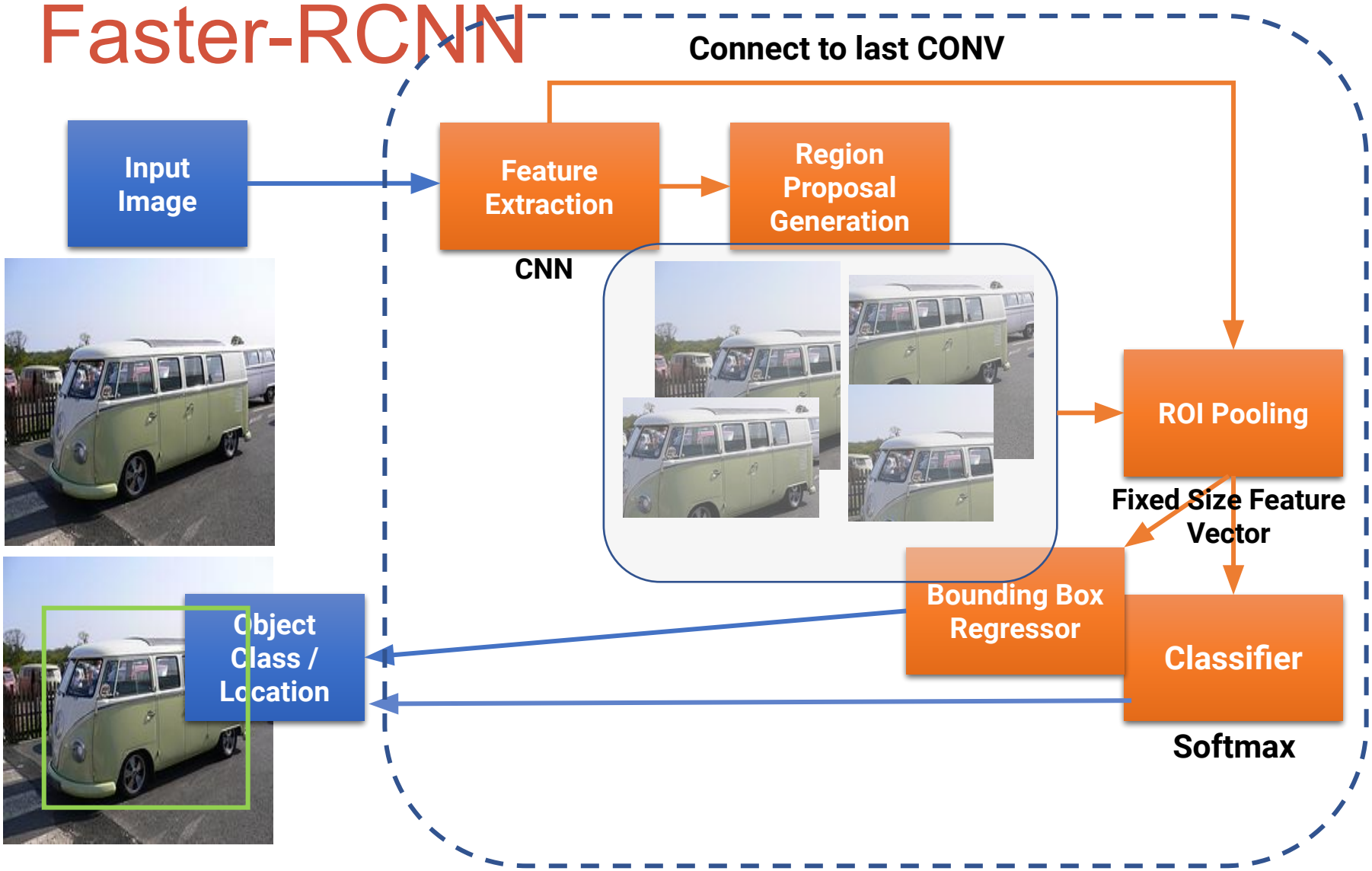
Faster-RCNN

- Combine the region proposal and classification into a single network
- End-to-End learning from region proposal to classification stage.



Shaoqing Ren, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks” 2015

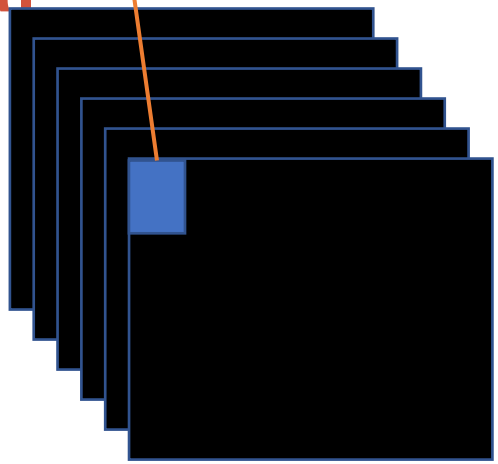
Faster-RCNN


Connect to last CONV
Input Image
Feature Extraction
Region Proposal Generation
CNN
ROI Pooling
Fixed Size Feature Vector
Bounding Box Regressor
Classifier
Softmax
Object Class / Location

Region proposal

Feature Extraction

3x3 sliding window



Last CONV output feature maps

1
2
3
5
7 8
9 0
1 5
..

Fixed Size feature vector

1
2
3
5
7 8
9 0
1 5

Region Proposal Regressor

Relative location (x,y) to image size

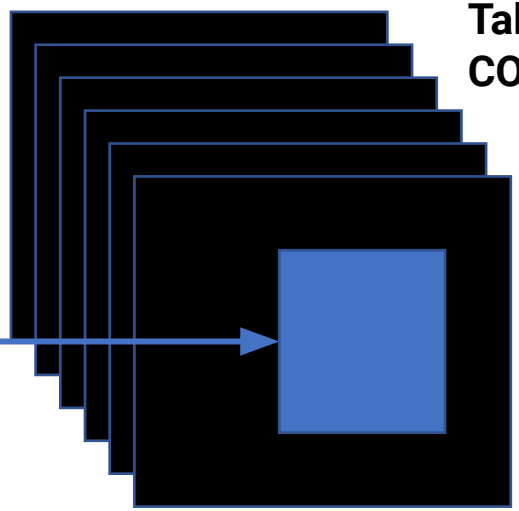
Objection Proposal Classifier

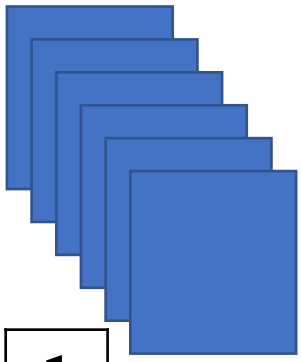
Object / Not Object

Final output

Feature Extraction

Take feature directly from CONV layer

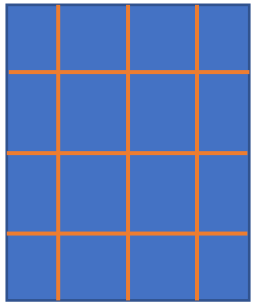




Corresponding location respect to region proposal location

Last CONV output feature maps

Region Proposal Generator



ROI Pooling: Divided into fixed partition and take max value

1
2
3
5
78
90
15
..

Fixed size feature vector

Mask R-CNN

Focal loss

Feature Pyramid Network

Joint loss

Issues with object detection

1) **Class imbalance** background class (no object) dominates the training data

2) **Scale** objects can be at different scale. Typical backbone handles one scale

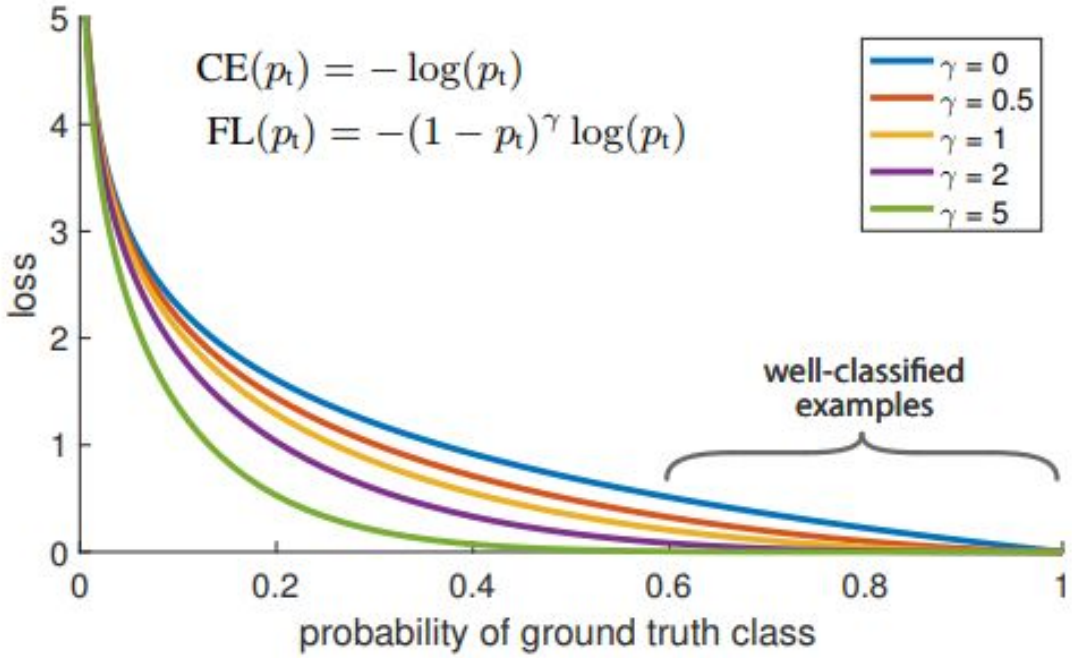
Focal Loss

Object detectors suffer from class imbalance

- Most boxes have no objects

Some training examples are also easy to classify

- Cross entropy put too much focus on easy examples



https://arxiv.org/pdf/1708.02002.pdf

Class weights to handle imbalance

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

P probability of predicting class 1

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$$

P_t Probability of being correct

$$\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t)$$

Add α which put emphasis on different class. It can be the inverse of number of training data for that class. (Class weights)

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t).$$

Help reduce class imbalance problem.

Focal loss

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

P probability of predicting class 1

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$$

P_t Probability of being correct

$$\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t)$$

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t).$$

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma} \log(p_t)$$

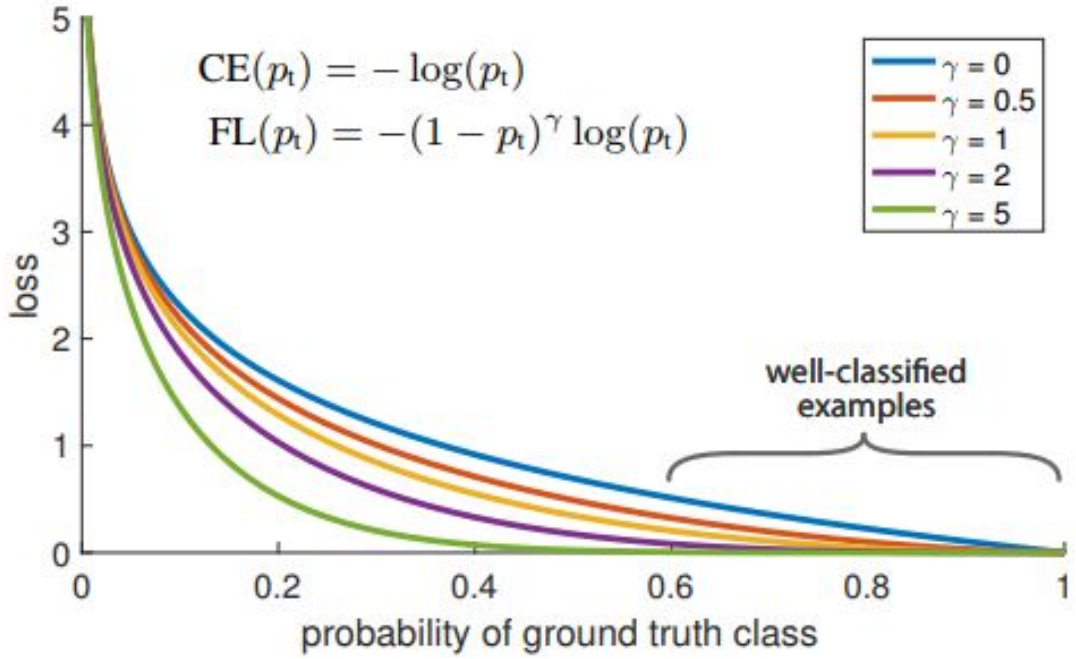
Focal loss

$$\mathrm{FL}(p_t) = -\alpha_t (1-p_t)^{\gamma} \log(p_t)$$

Focal loss with class weights

Focal Loss

Focal loss put higher emphasis on harder to classify examples



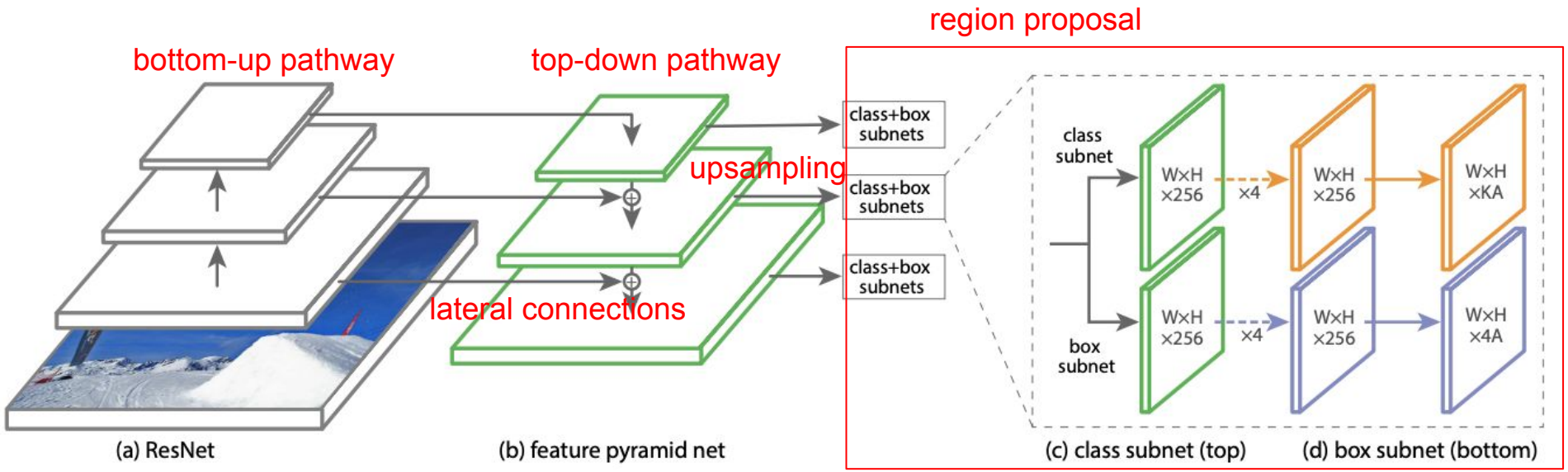
Good for many tasks besides CV

https://arxiv.org/pdf/1708.02002.pdf

Multi-scale with feature pyramid network

Handle multi-scale by combining different layers of the network

Add lateral connections (1x1 conv)



Focal Loss for Dense Object Detection, 2017

Mask R-CNN

Combines segmentation with object detection to jointly learn (instance segmentation)

Mask prediction is binary classification on the RoI region

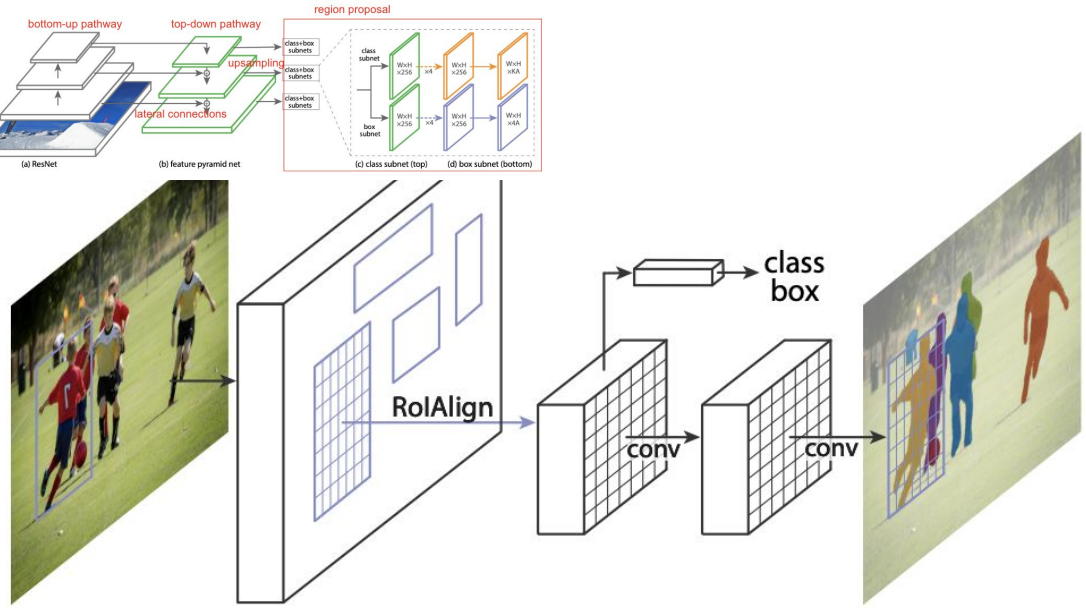
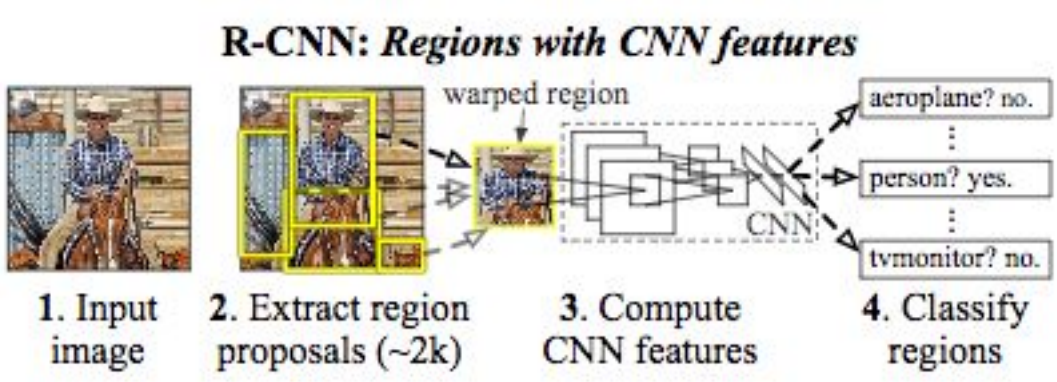


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Mask R-CNN, 2018

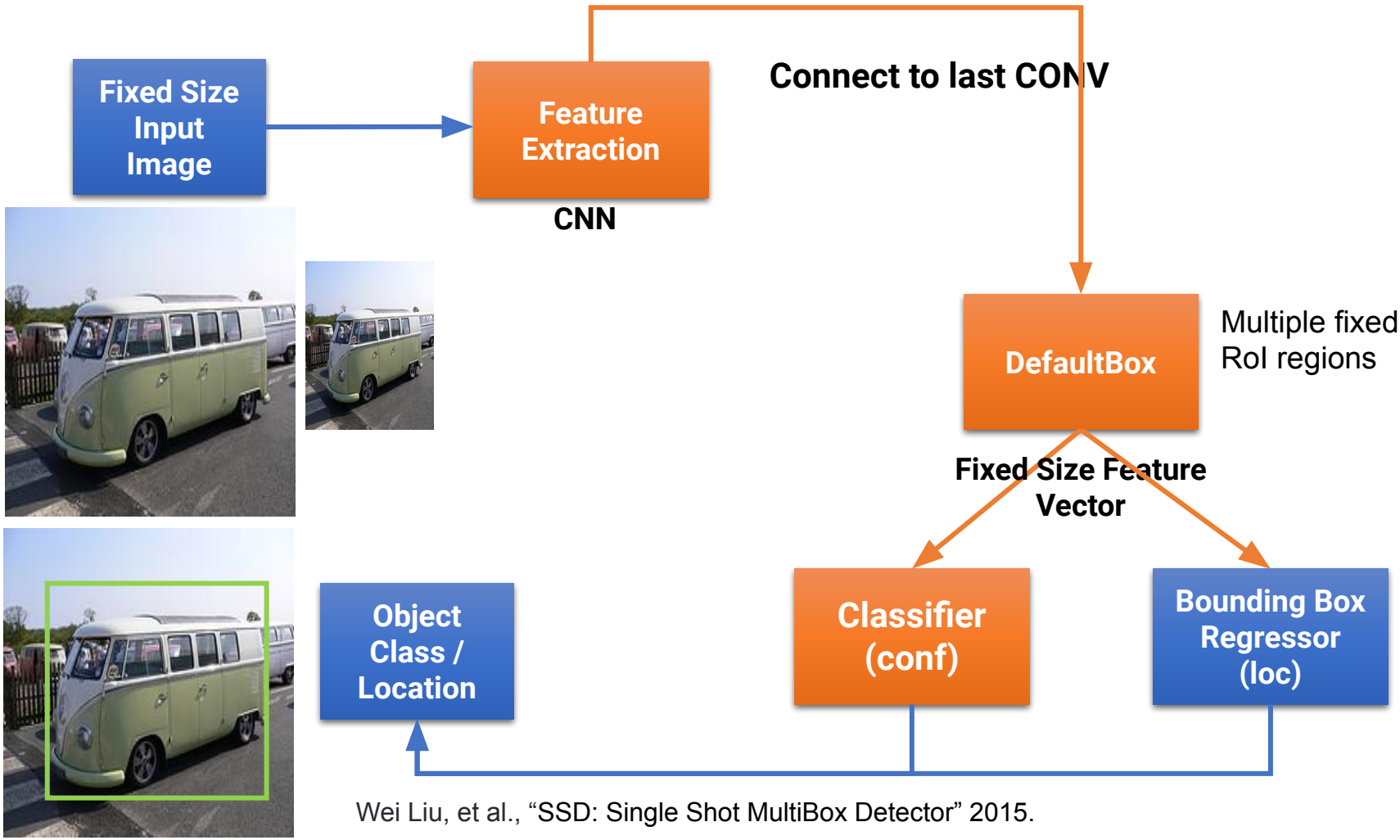
Object Detection Comprehensive Review

1. Proposal-based detector
   - R-CNN
   - Fast-RCNN
   - Faster-RCNN
   - Mask RCNN
2. Single stage detector
   - YOLO
   - SSD



SSD : Single Shot Multibox Detector

- Trade off accuracy for computational speed => Realtime detector

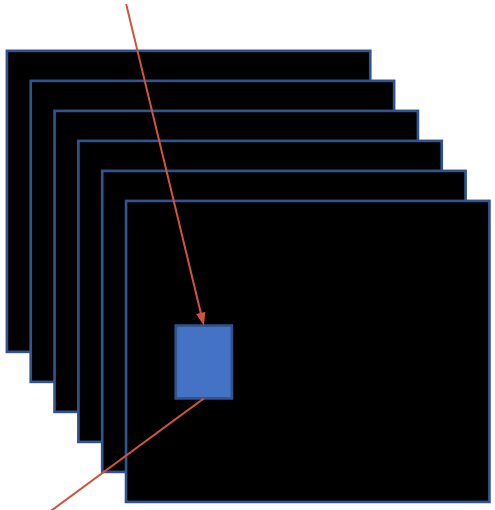


Wei Liu, et al., “SSD: Single Shot MultiBox Detector” 2015.

SSD : Default Box Idea

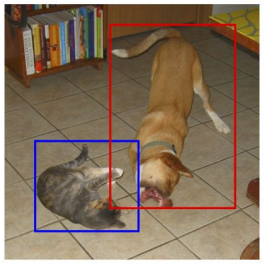
Feature Extraction

3x3 conv

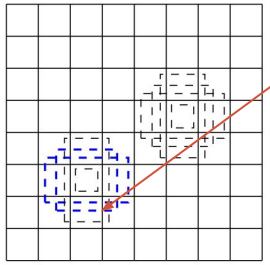


Last CONV output feature maps

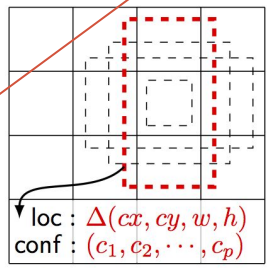
In training process, SSD will optimize loc,conf which corresponds to the object's real position based on IOU criterion.



(a) Image with GT boxes



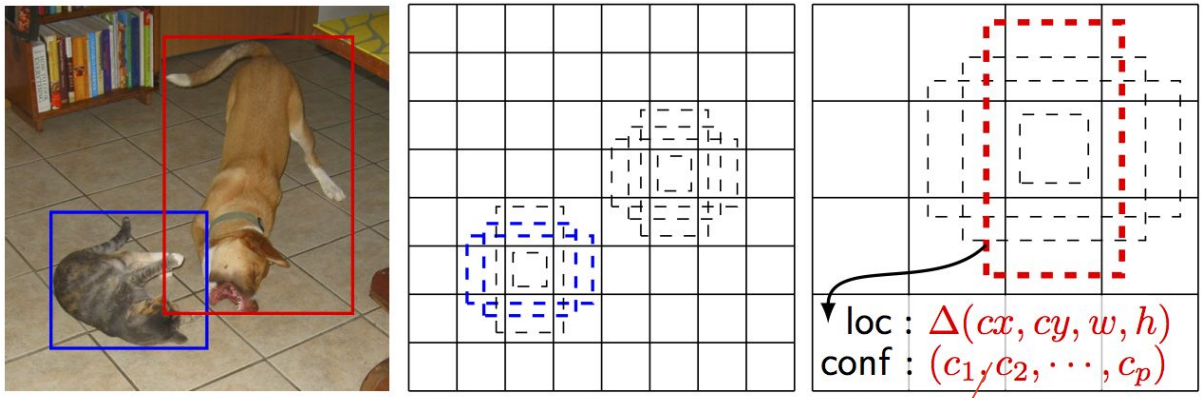
(b) 8×8 feature map



(c) 4×4 feature map

Every feature map cell is associated with a set of default bounding boxes of different dimensions and aspect ratios. Represents "default box" in xywh coordinate system (loc) and class output (conf)

SSD



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

$$MultiBoxLoss(x, c, l, g) = L_{conf}(CrossEntropyLoss(x, c)) + L_{loc}(SmoothL_1Loss(x, l, g))$$

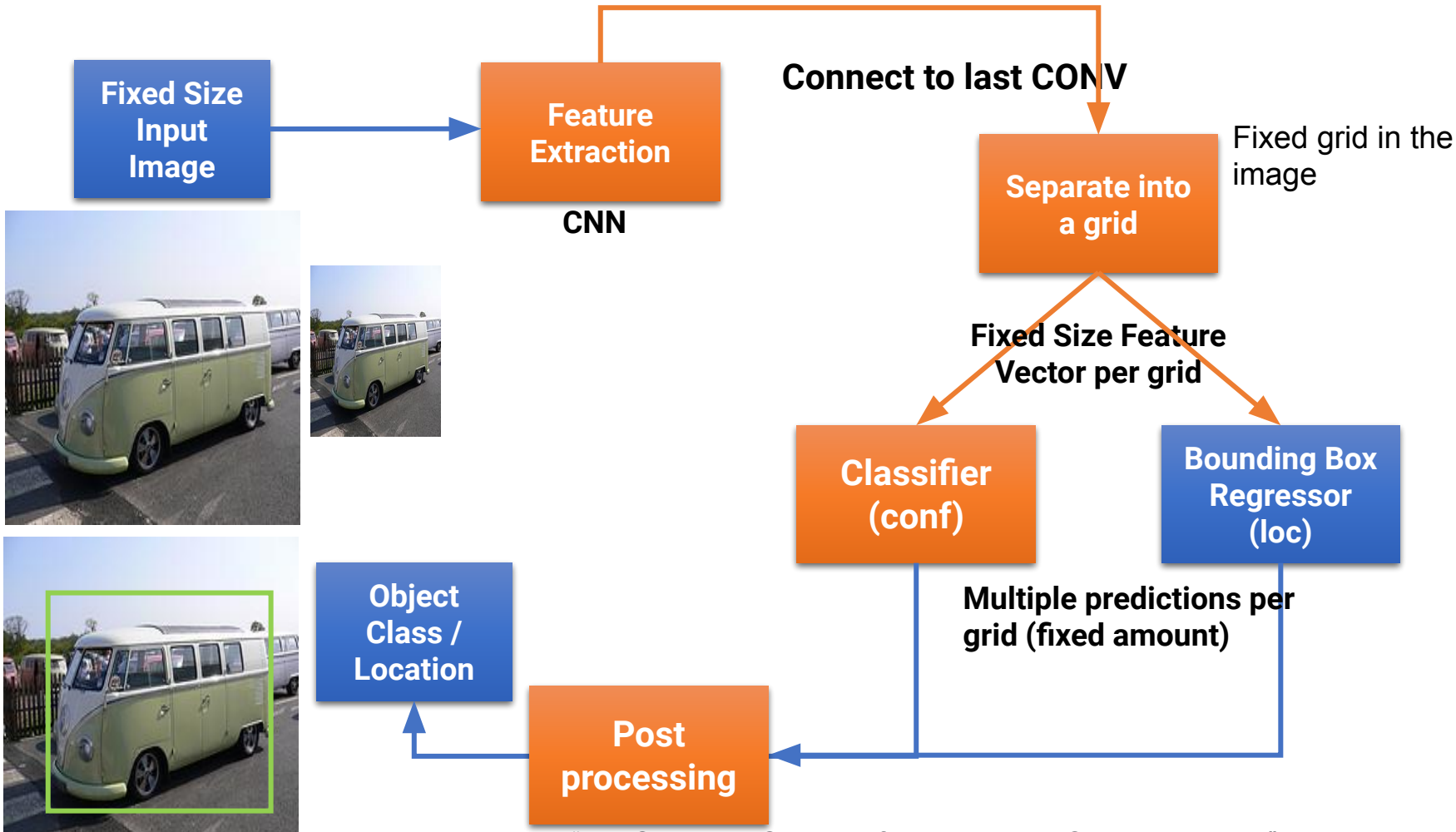
IOU > 0.5 (Match with O1)

IOU > 0.5 (Match with O2)

SSDLoss = Classification loss + Localization Loss

YOLO: You Only Look Once

- Similar concept as SSD, fixed locations to look for things



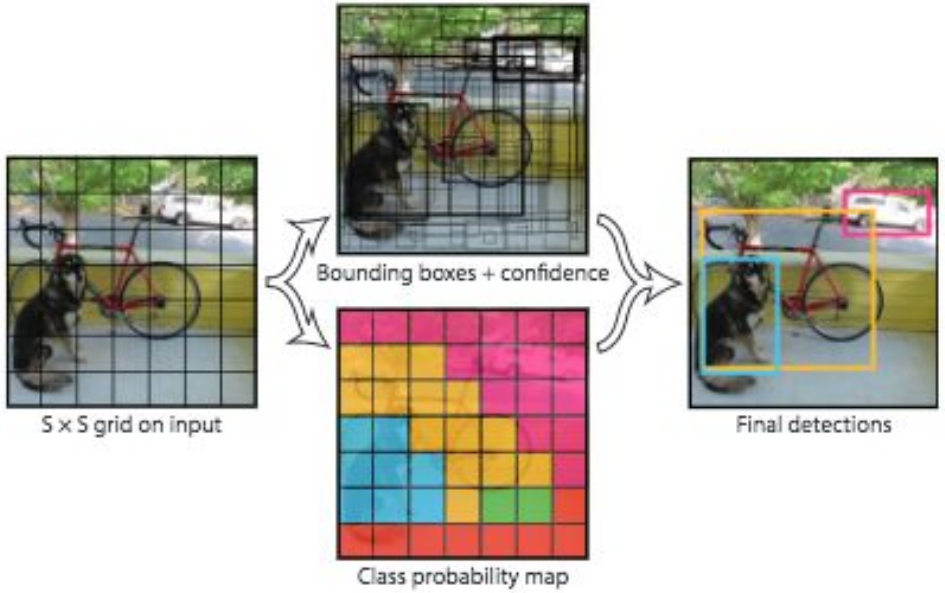
Joseph Redmon, "You Only Look Once: Unified, Real-Time Object Detection," 2015

YOLO (You Only Look Once)

SSD have multiple boxes per location (multi-resolution), YOLO has one box per location

YOLO makes multiple predictions per box at once.

Limitation of YOLO: cannot predicts things that are close together.



YOLO v2 (YOLO9000)-v3

Batch norm

Anchor boxes: fixed set of bounding boxes per grid

Handles multi-resolution by concatenating features from multiple parts of the backbone

Joseph Redmon, et al. “YOLO9000: Better, Faster, Stronger” 2016
Joseph Redmon, et al. “YOLOv3: An Incremental Improvement” 2018

Misc topics

- Encoder-Decoder
- Attention modeling
- Transfer Learning, Multi-task models, Multi-models
- Object detection

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

Course project

- 4 people
- Topic of your choice
  - Can be implementing a paper
  - Extension of a homework
  - Project for other courses with an additional machine learning component
  - Your current research (with additional scope)
  - Or work on a new application
  - Must already have existing data! No data collection!
- Topics need to be pre-approved

Projects idea

Clock drawing for Parkinson diagnosis

Project doodle

- Build your group on courseville (under project)
- Submit a proposal. Due next Tuesday.
  - Bullet points
  - What do you want to do?
  - What is the data?
  - How to evaluate the task?
- Sign up for time slots (next week's office hour)
  - Sign up sheet on courseville (as an assignment)