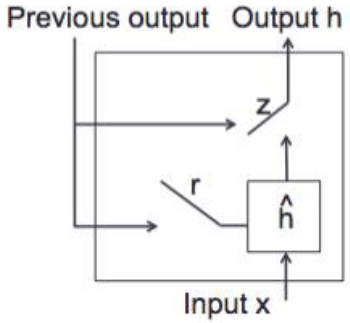
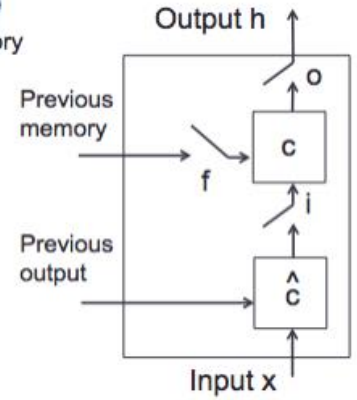
Transformers

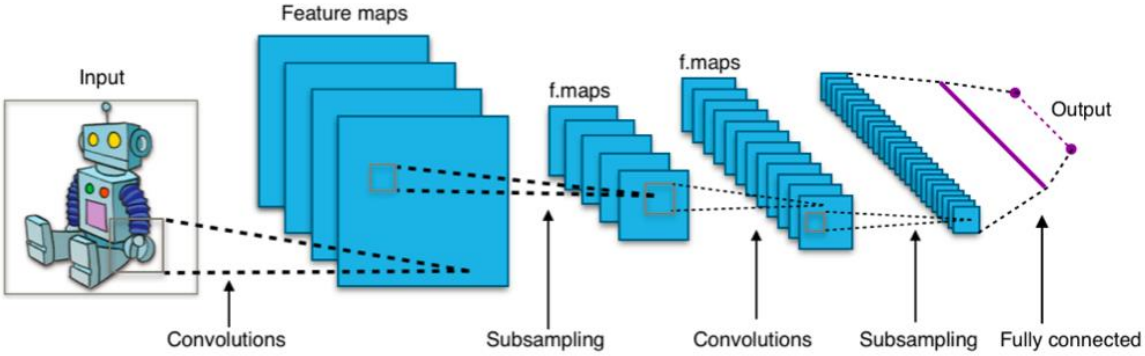
Review from last time

- CNN
- RNN+LSTM

- Has an explicit memory cell (c)
  - Does not have to output the memory



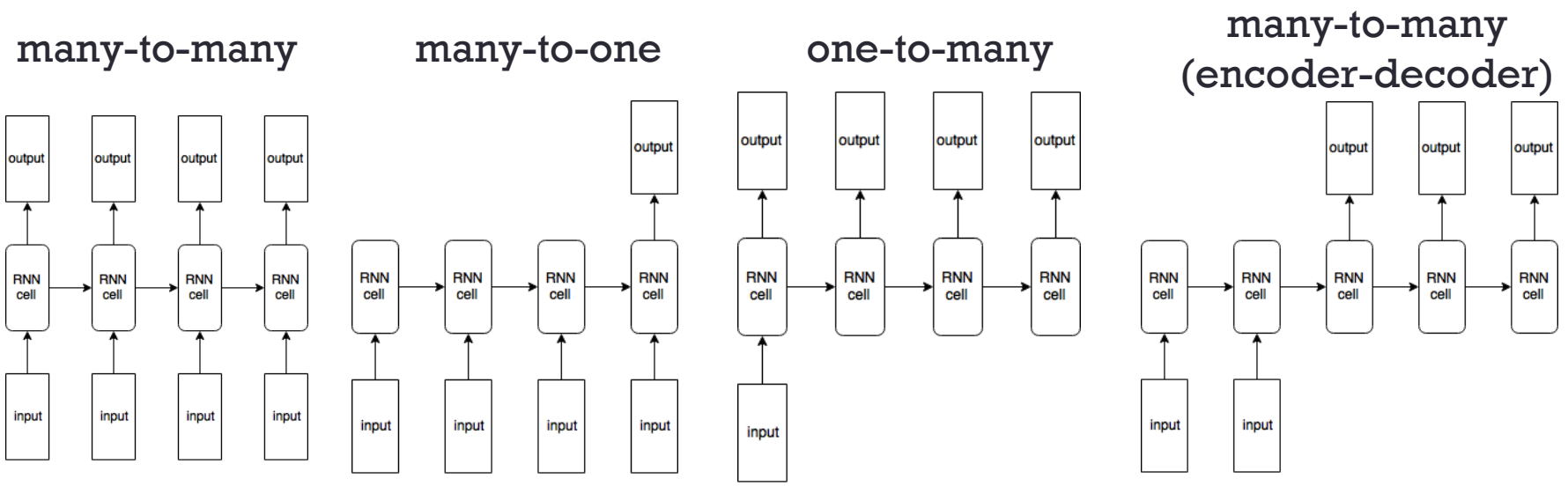




Structure of this lecture

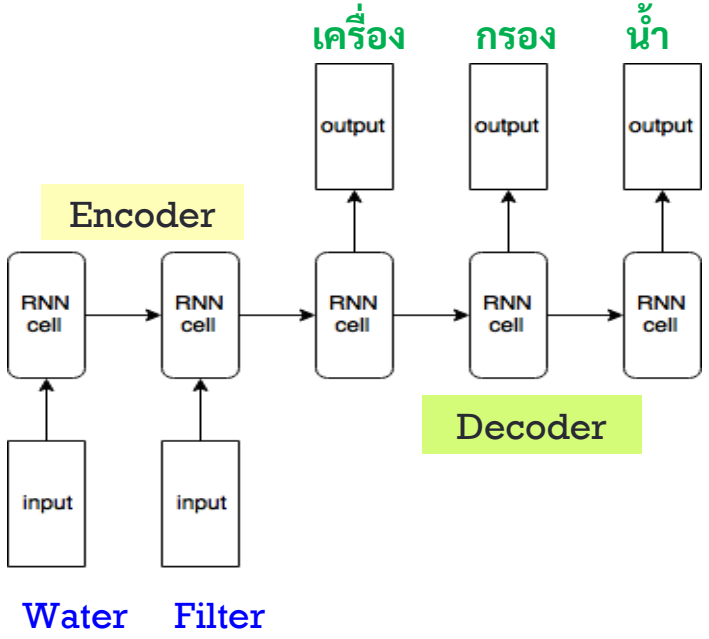
- Attention
- Transformer
- Transformer is all you need?

Different recurrent architectures



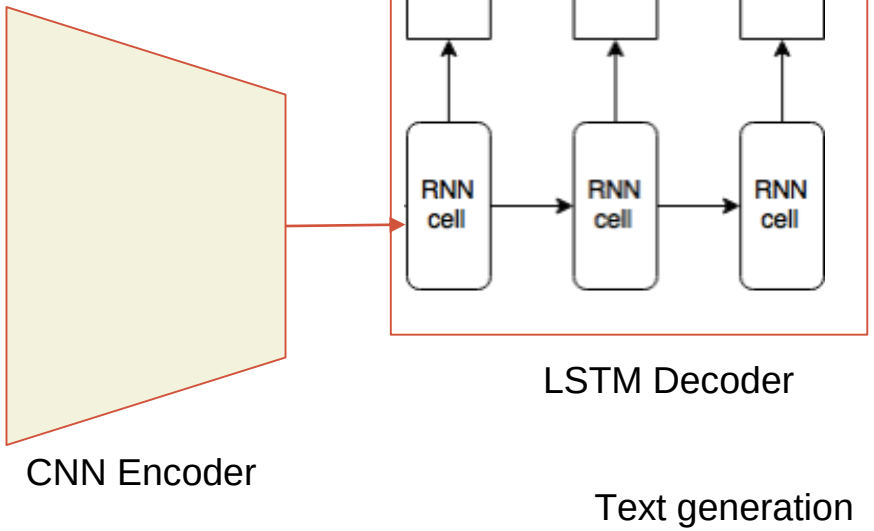
Many-to-many (encoder-decoder)

- Sequence Input, Sequence output
- These two sequences can be of different length
- E.g. Machine Translation
  - Input: English Sentence
  - Output: Thai Sentence
- Encoder-decoder can be viewed as more than just text
- **Let's look at the decoder part first (generation)**



Encoder-decoder

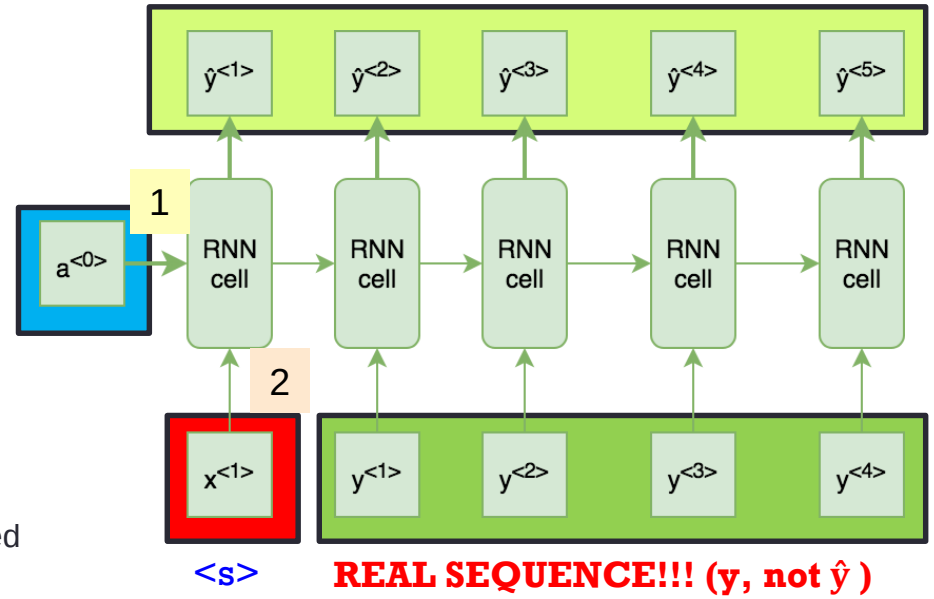
- Input and output can be of different modality
  - Speech, text, image, etc.
- Speech to image (image generation)
- Image to text (image description)
- Image to code
  - https://sketch2code.azurewebsites.net/



Training Inference

Text generation model (training)

- **One-to-Many RNN (autoregressive)**
  - 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.

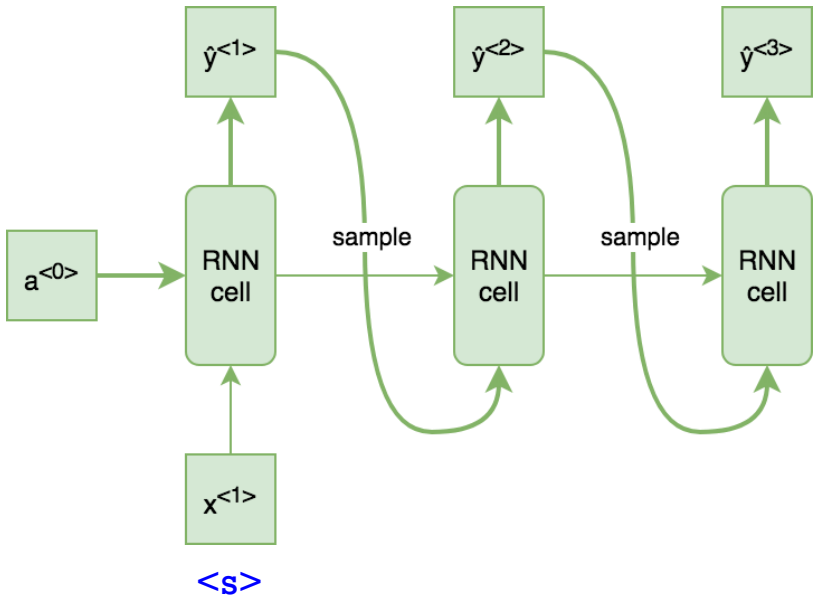


$$a^{<t>} = \underbrace{W a^{<t-1>}}_{1} + \underbrace{W x^{<t>}}_{2} + b$$

Training Inference

Text generation model (inference; testing)

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



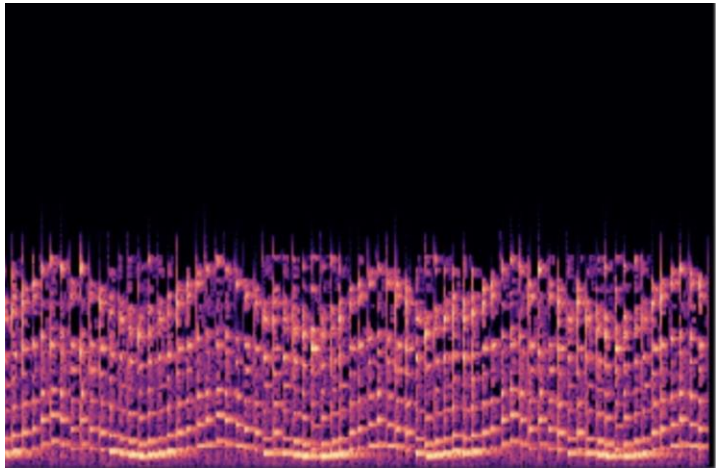
Text generation example

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

Input: spectrum
Predict: spectrum

https://soundcloud.com/661_25/generation-no-60-master-cnn

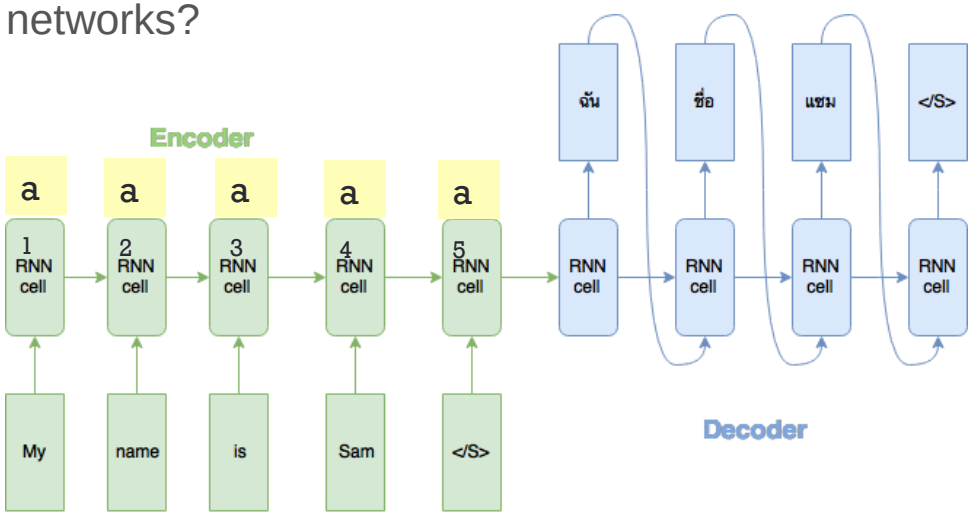
https://noppawat-tan.medium.com/thai-music-generation-ranat-ek-with-variational-autoencoder-753e69f0b323

Attention Mechanism (Many-to-Many)

Attention is commonly used in sequence-to-sequence model, it allows **the decoder part** of the network to focus/**attend** on a different part of **the encoder outputs** for every step of the decoder's own outputs.

Why attention?

This is what we want you to think about: How can information travel from one end to another in neural networks?

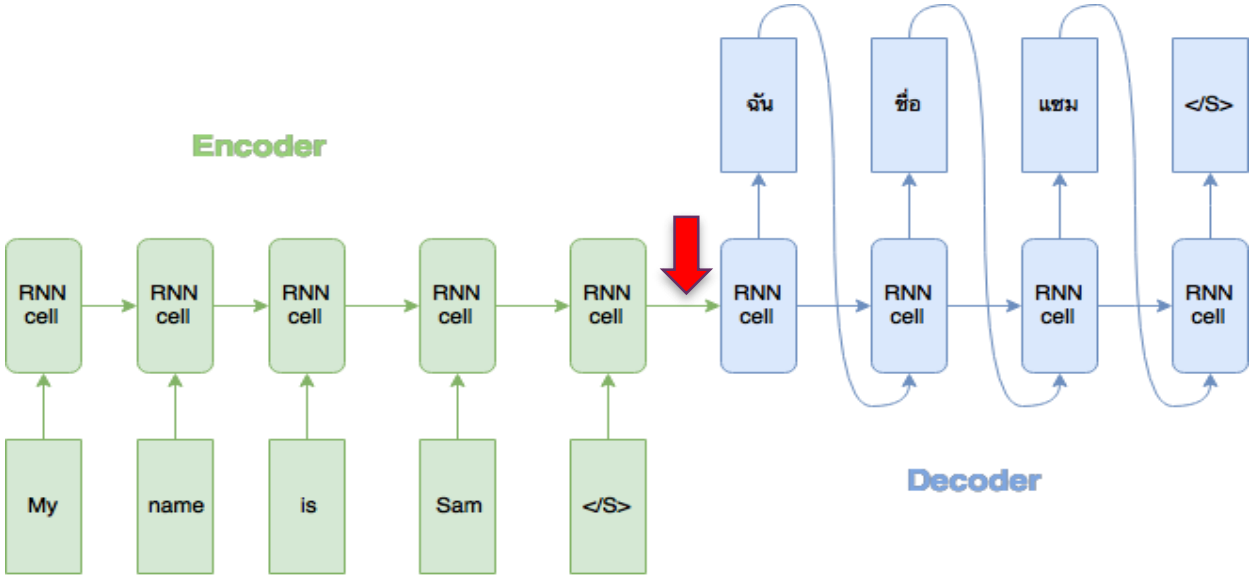


Machine Translation Problem: English to Thai

Attention Mechanism (cont.)

Why attention?

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!" - Raymond Mooney (2014)



Machine Translation Problem: English to Thai

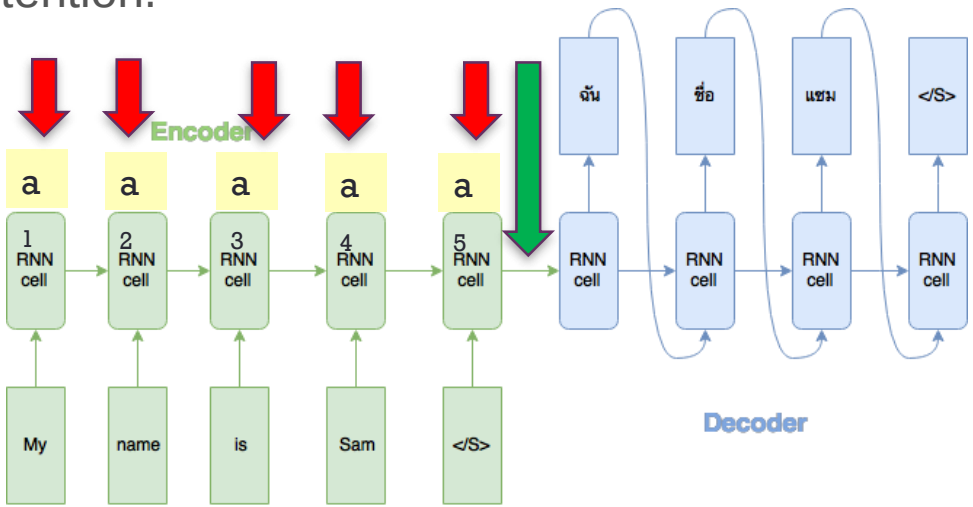
Reference:http://yoavartzi.com/sp14/slides/mooney.sp14.pdf

Attention Mechanism (cont.)

Why attention?

Main idea: We can use multiple vectors based on the length of the sentence instead of one.

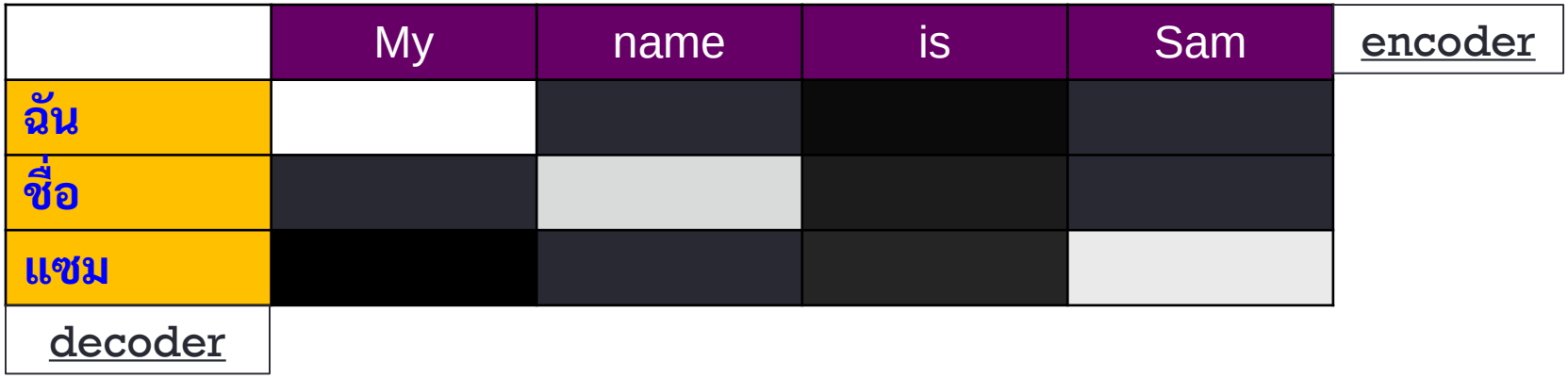
Attention mechanism = Instead of encoding all the information into a fixed-length vector, the decoder gets to decide parts of the input source to pay attention.

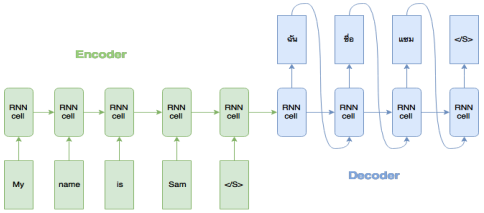


Machine Translation Problem: English to Thai

Graphical Example: English-to-Thai machine translation

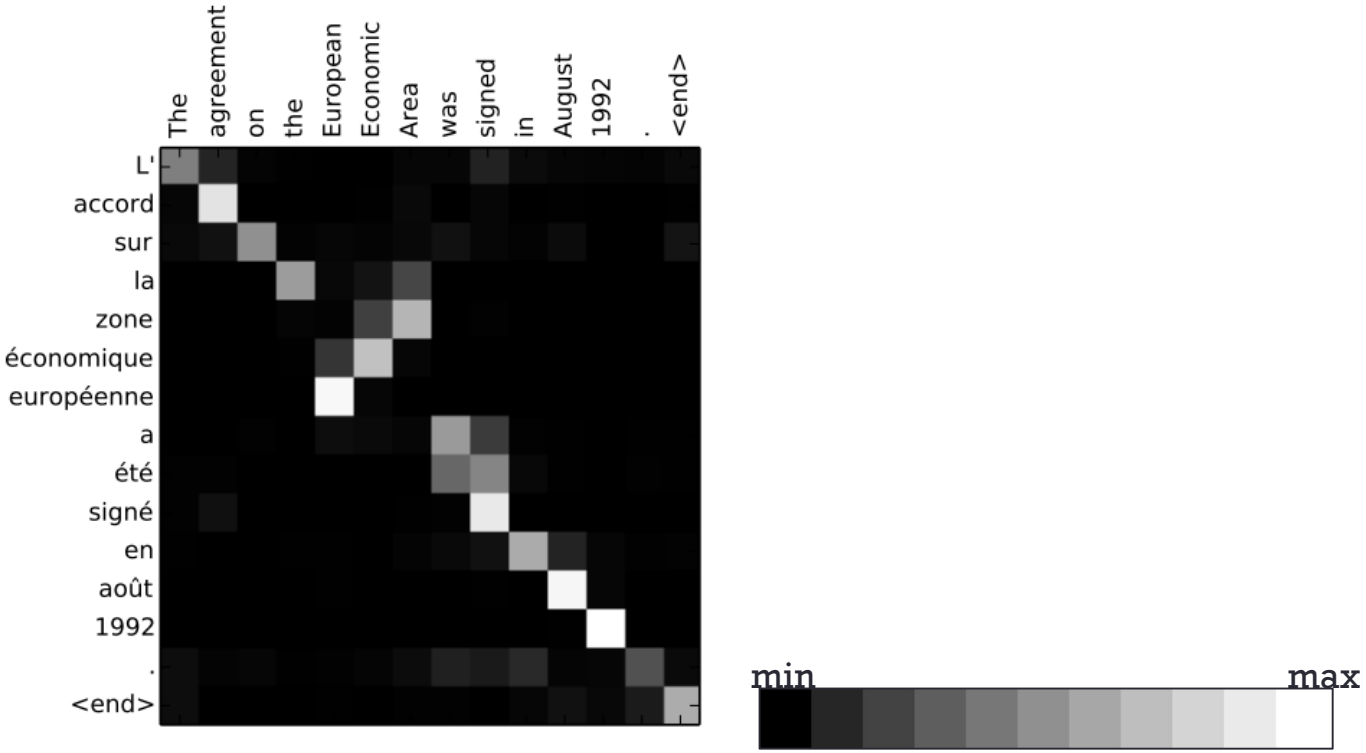
- This is a rough estimate of what might occur for English-to-Thai translation





Machine Translation Problem: English to Thai

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).

Attention Mechanism: Basic Idea

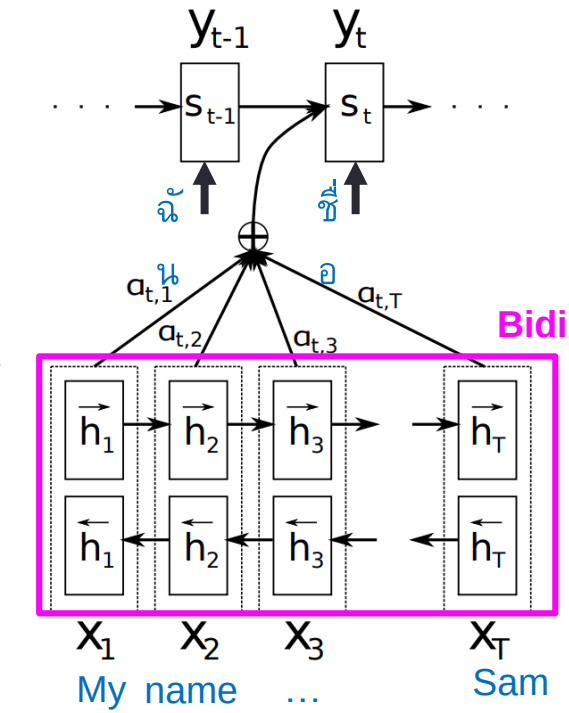
- Encode each word in the sequence into a vector
- When DECODING, perform a linear combination of these encoded vectors from the encoding step with their corresponding “attention weights”.

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

j = each encoder’s input
i = each decoder’s input

- A vector formed by this linear combination is called “context vector”
- Use context vectors as inputs for the decoding step

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



Decoder

Bidirectional LSTM

Encoder

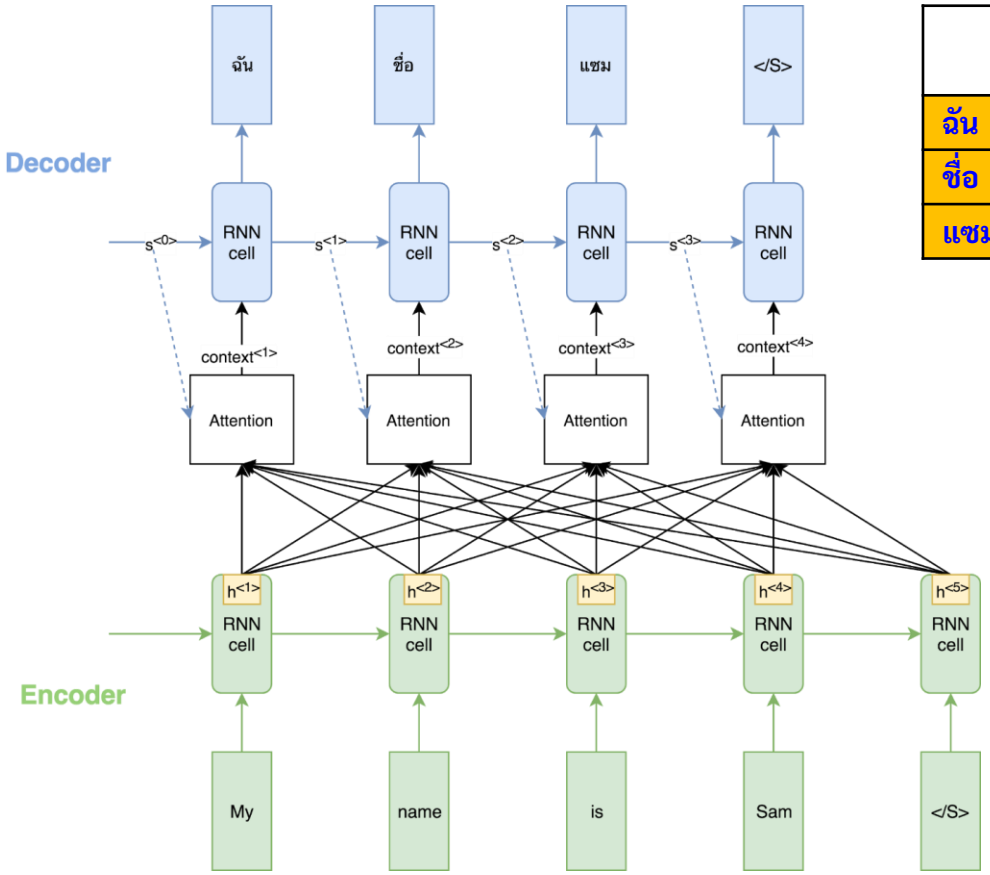
Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

target word = decoder

source = encoder

Reference: Bahdanau, D., Cho, K., & Bengio, Y.. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

RNN and attention mechanism



	My	name	is	Sam
ฉัน				
ชื่อ				
แซม				

Attention Mechanism (1): C_i

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 $\mathbf{a}_i$.

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

context vector

attention score

encoder state at index j

i = decoder index
j = encoder index

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

previous hidden state/
decoder state

attention score(weight vector)
of encoder state at index j

encoder state at index j

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

Attention Mechanism (2): f_{att}

The attention function $f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j)$ calculates an unnormalized alignment score between the current hidden state $\mathbf{s}_{i-1}$ and the previous hidden state $\mathbf{h}_j$.

There are many variants of the attention function f_{att}.

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

context vector

attention score

encoder state at index j

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

previous hidden state/ **decoder** state

attention score(weight vector) of encoder state at index j

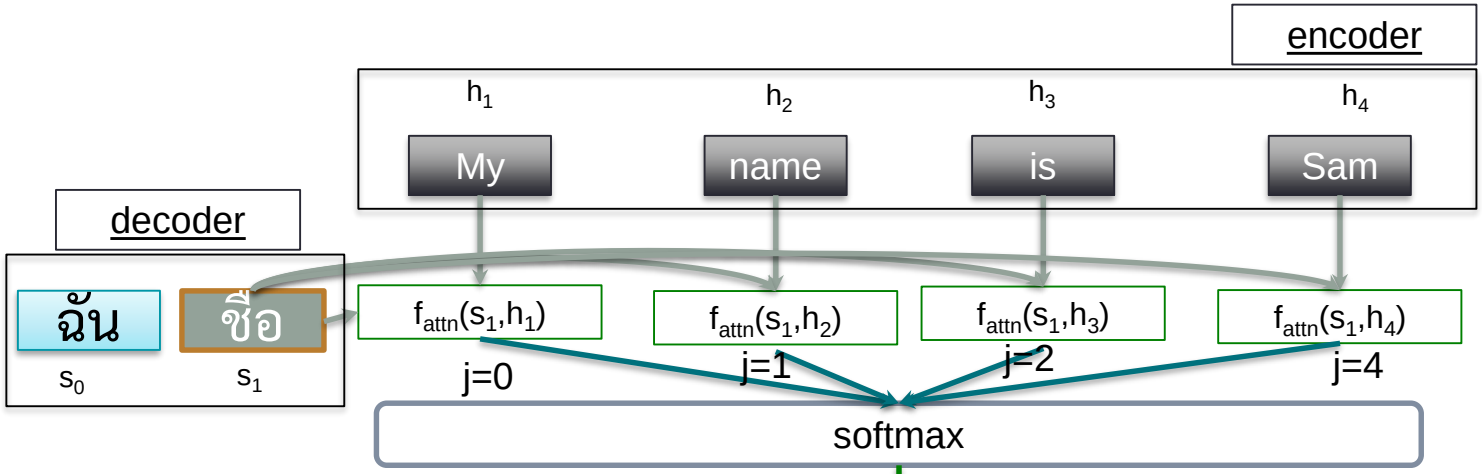
encoder state at index j

i = decoder index
j = encoder index

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

Attention Calculation Example (1): Attention Scores

Now we want to predict แซม (i=2)



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

$\mathbf{a}_2$=[$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ $a_{2,4}$]

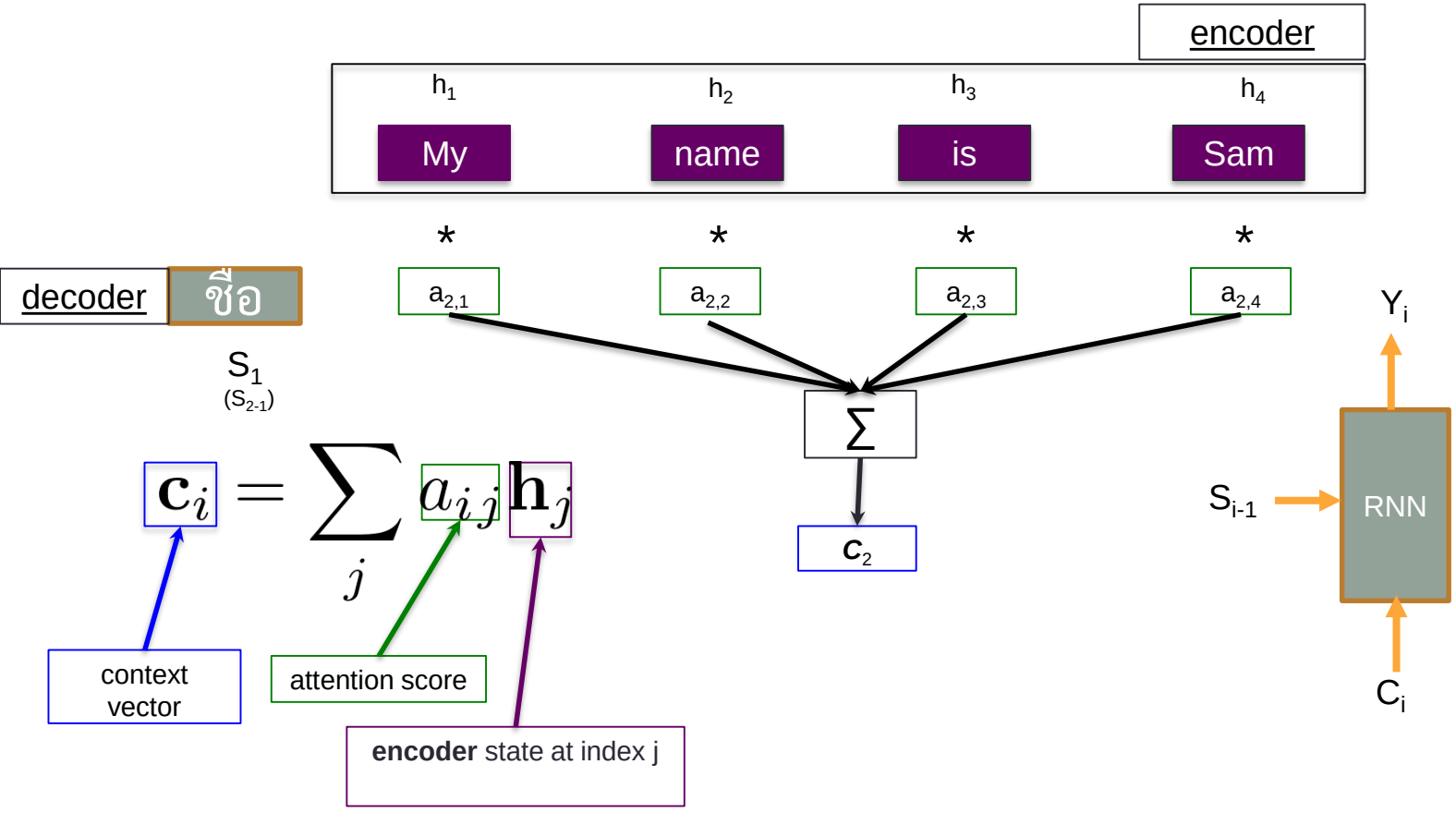
i = decoder index
j = encoder index

attention scores

previous hidden state/ decoder state

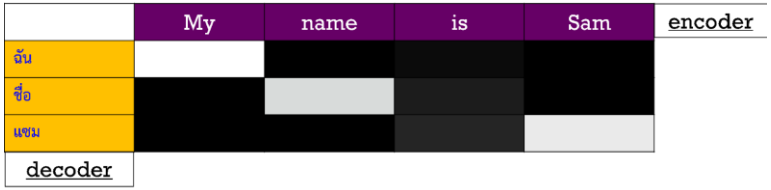
encoder state at index j

Attention Calculation Example (2): Context Vector



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

Type of Attention mechanisms



(Remember that 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)

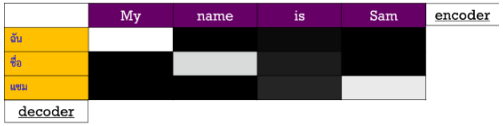
$$\mathbf{a} = softmax(\mathbf{w}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T))$$

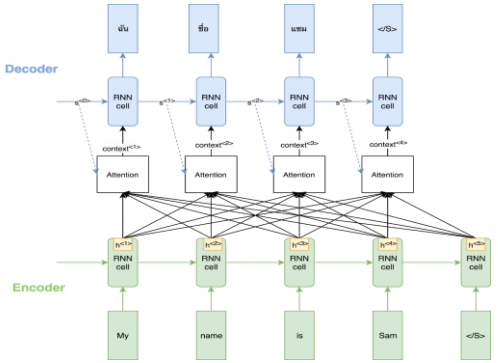
Key-value attention: key-value attention (Daniluk et al., 2017) is a recent attention variant that separates form from function by keeping separate vectors for the attention calculation.

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

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

Additive Attention





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

concatenation

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

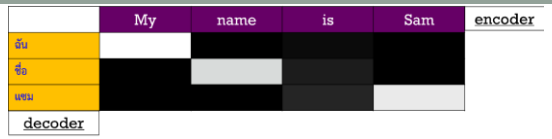
One-hidden layer (Dense)

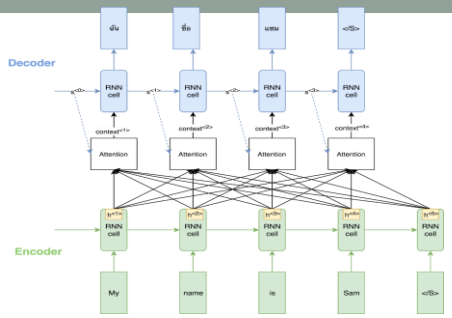
- Where $\mathbf{W}_a$ are learned attention parameters. Analogously, we can also use matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ to learn separate transformations for $\mathbf{s}_{i-1}$ and $\mathbf{h}_j$ respectively, which are then summed (hence the name additive):

$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = tanh(\mathbf{W}_1\mathbf{s}_{i-1} + \mathbf{W}_2\mathbf{h}_j)$$

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

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





Multiplicative Attention

- Multiplicative attention (Luong et al., 2015) [16] simplifies the attention operation by calculating the following function:

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

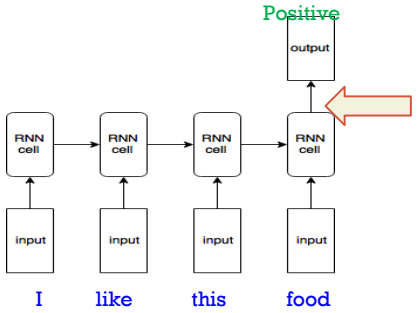
- Faster, more efficient than additive attention BUT additive attention performs better for larger dimensions
- One way to mitigate this is to scale f_{attn} by $\frac{1}{\sqrt{d_s}}$

d_s = #dimensions of hidden states in LSTM (context vector; latent factors)

  - Dot product of high dimensional vectors has high variance -> softmax is peaky -> small gradient -> harder to train

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

Self Attention (1)



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

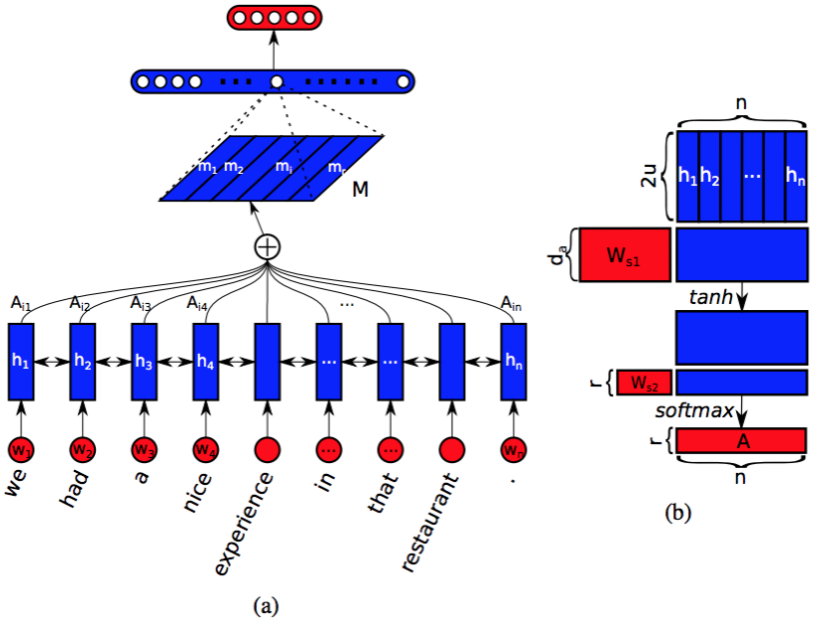
$$H = (\mathbf{h_1}, \mathbf{h_2}, \cdots \mathbf{h_n})$$

Fully connected layer

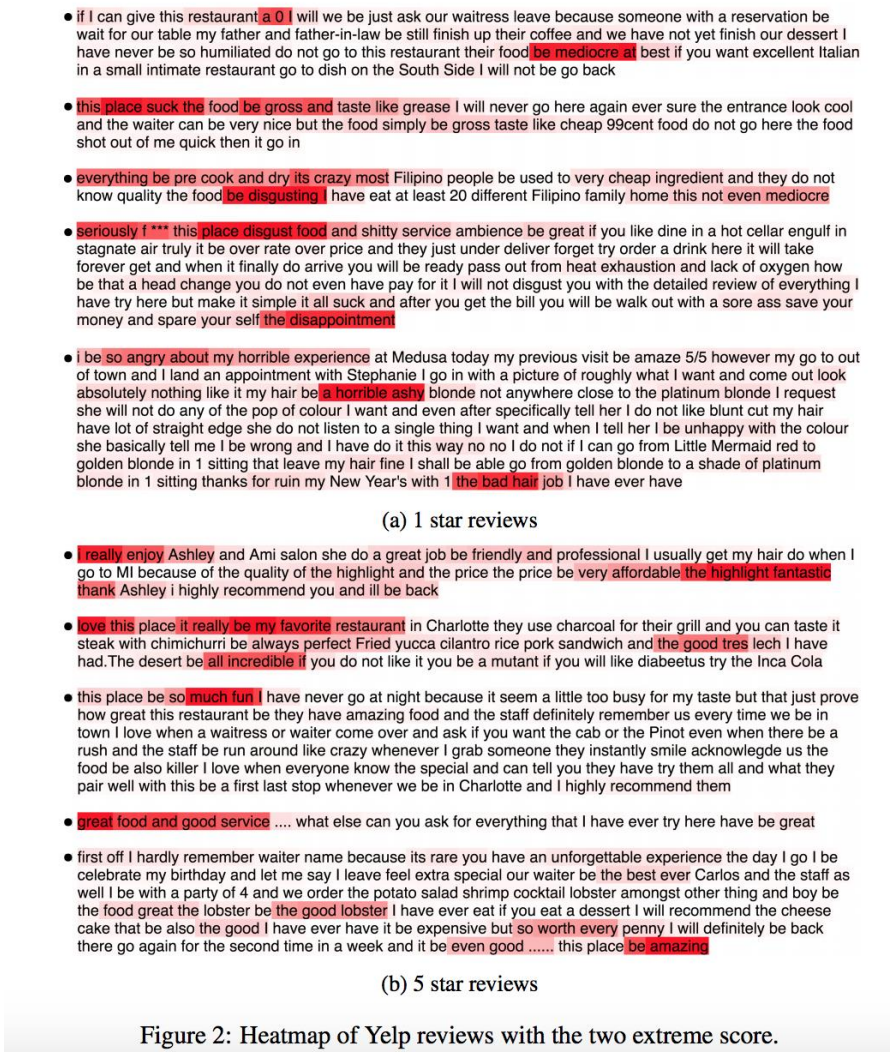
$$\mathbf{a} = softmax(\mathbf{w}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T))$$

One-hidden layer (Dense)

- w_{s1} is a weight matrix, w_{s2} is a vector of parameters. Note that these parameters are tuned by the neural networks.
- **The objective is to improve a quality of embedding vector by adding context information.**



Self-attention (2)



- 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

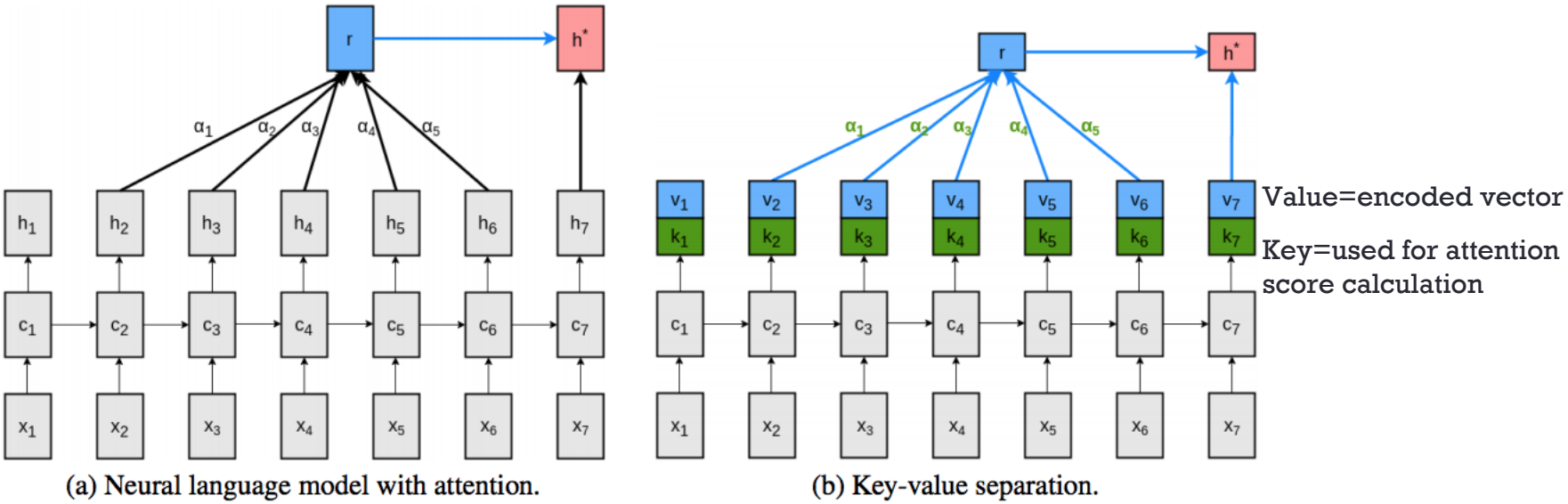
(b) 5 star reviews

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

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

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

Key-value attention (1)



(a) Neural language model with attention.

(b) Key-value separation.

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

Key-value attention (2)

previous outputs (memory) **with window L**

current output

k - output dimension
L - output length

key

value

$$\begin{bmatrix} \boldsymbol{k}_t \\ \boldsymbol{v}_t \end{bmatrix} = \boldsymbol{h}_t \quad \in \mathbb{R}^{2k}$$

$$\boldsymbol{M}_t = \tanh(\boldsymbol{W}^Y [\boldsymbol{k}_{t-L} \cdots \boldsymbol{k}_{t-1}] + (\boldsymbol{W}^h \boldsymbol{k}_t)\mathbf{1}^T) \quad \in \mathbb{R}^{k \times L}$$

Attention score

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{w}^T \boldsymbol{M}_t) \quad \in \mathbb{R}^{1 \times L}$$

Context vector (v_t not included)

$$\boldsymbol{r}_t = [\boldsymbol{v}_{t-L} \cdots \boldsymbol{v}_{t-1}]\boldsymbol{\alpha}^T \quad \in \mathbb{R}^k$$

The final representation

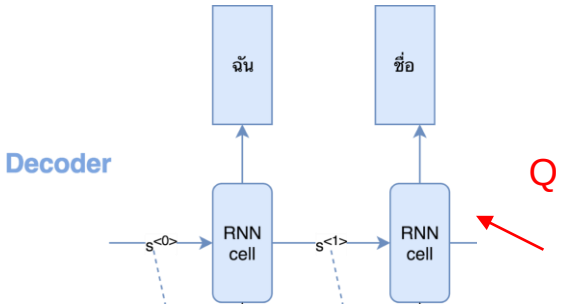
$$\boldsymbol{h}_t^* = \tanh(\boldsymbol{W}^r \boldsymbol{r}_t + \boldsymbol{W}^x \boldsymbol{v}_t) \quad \in \mathbb{R}^k$$

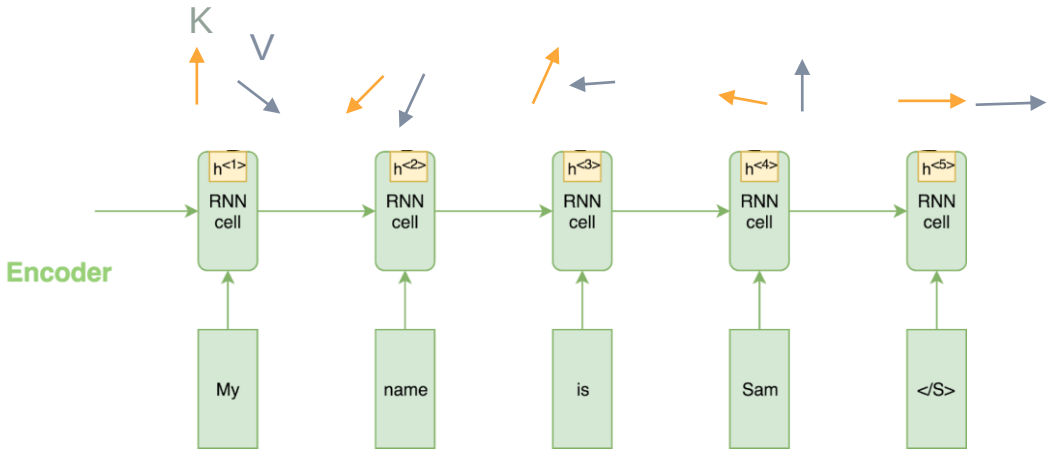
Context vector (v_t not included)

Original vector at index t

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

Pictorial view of KV attention



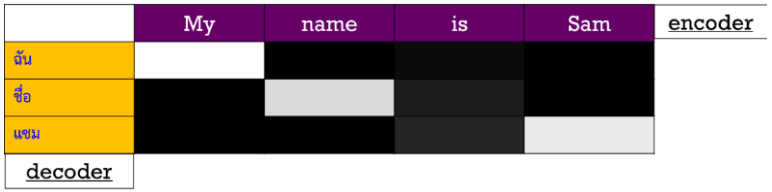


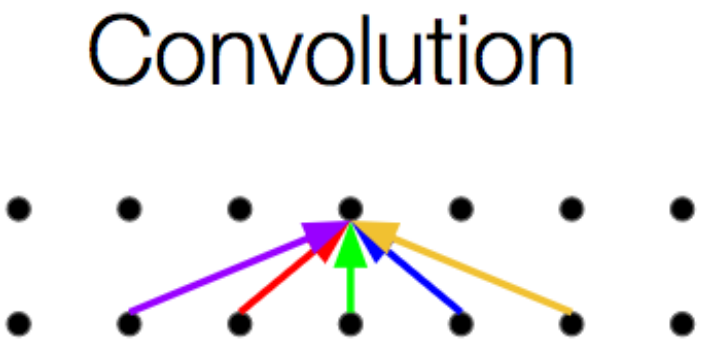
Convolution vs Self attention

Limited context vs Wide context

Downside: $O(n^2)$ when computing attention weights

Fixed weights vs weights changes according to input





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

Structure of this lecture

- Attention
- Transformer
- Transformer is all you need?

Transformer

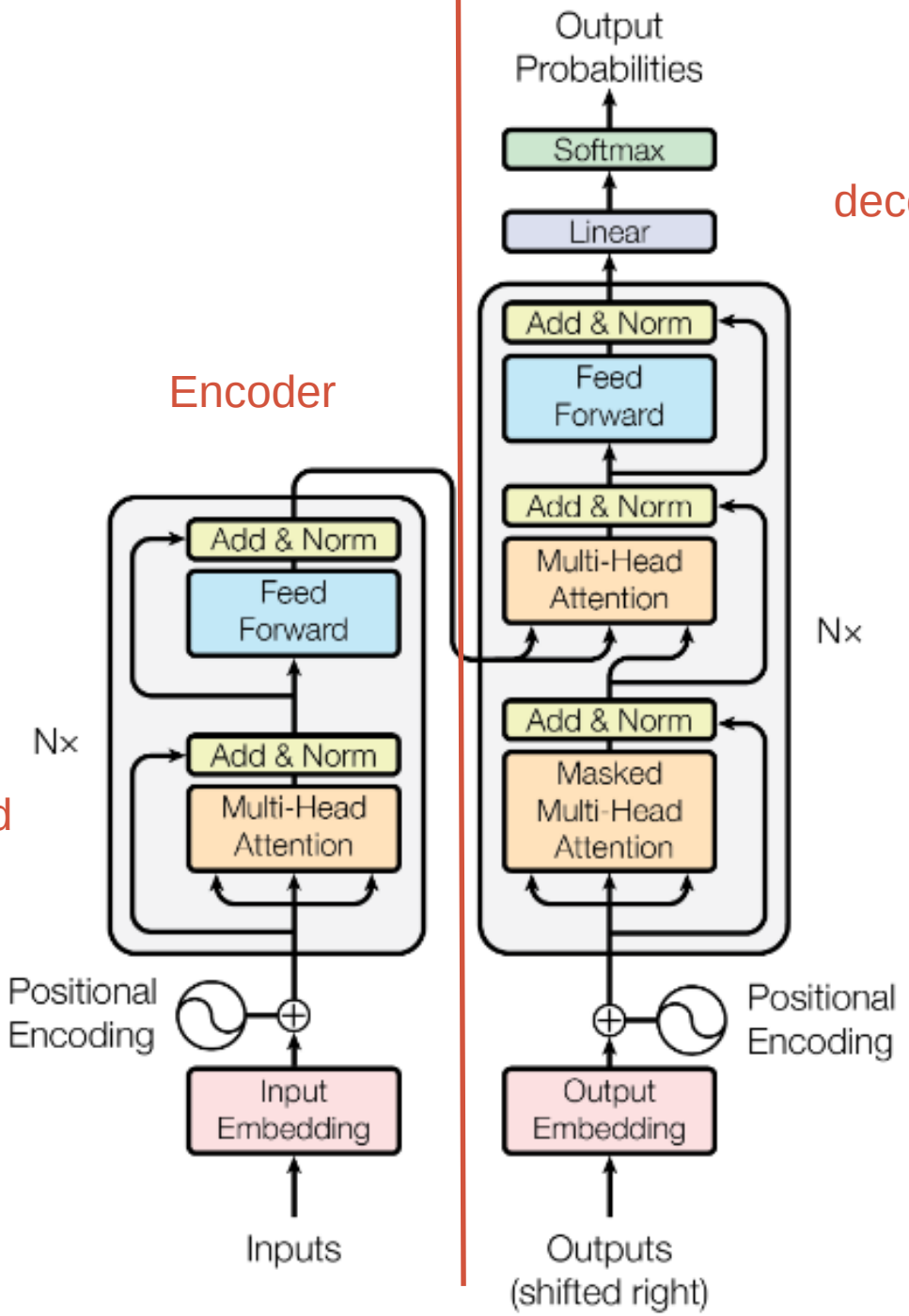
Attention is all you need

Abstract

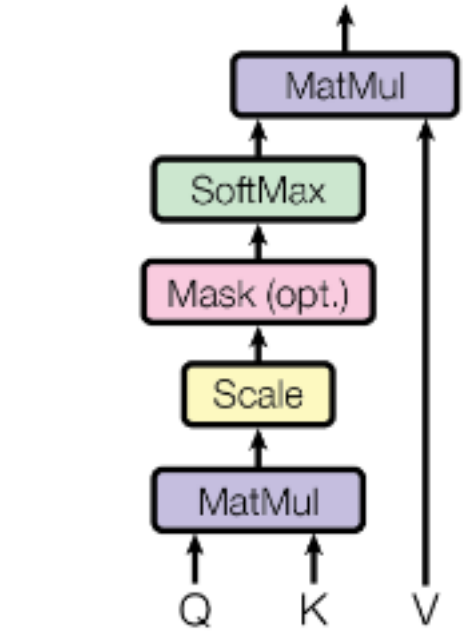
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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.

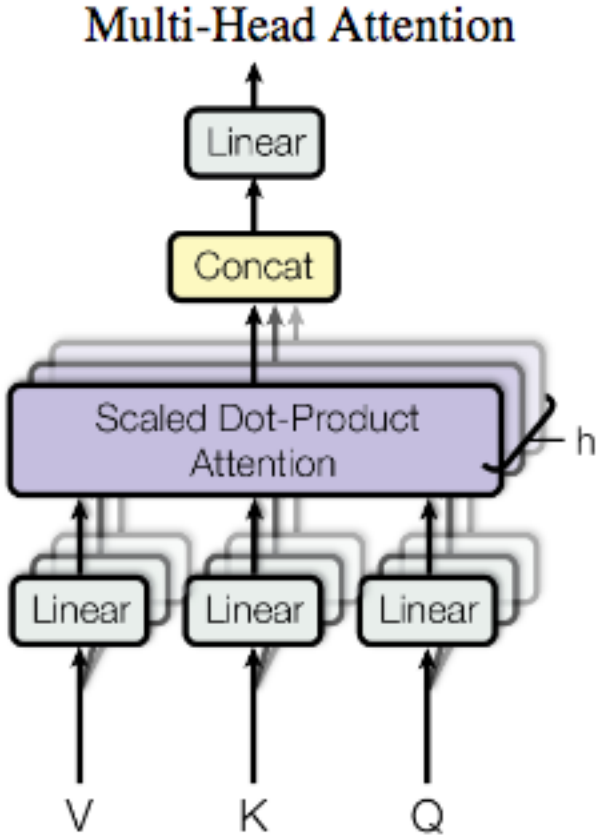
https://arxiv.org/pdf/1706.03762.pdf

To eliminate GRU which remembers time, encode position instead



Multi-head attention





What's this????

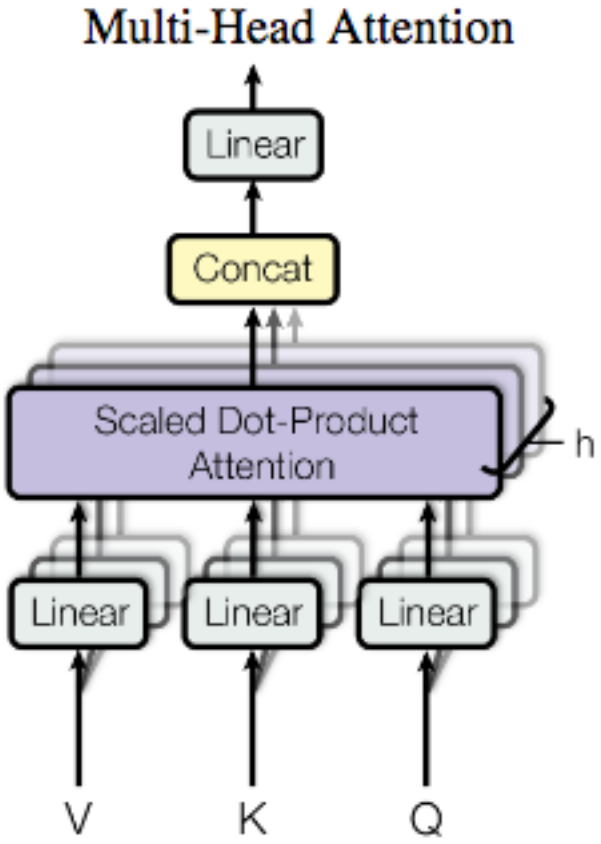
Query – used with Key to determine the position
Value – used as the information after determining the position

Attention drawback

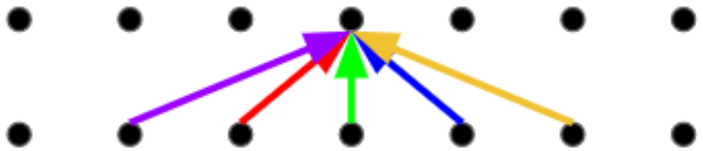
- Convolution: weights * input. Each weights are different. So position is encoded.
- Self-attention: a weighted average. Position information is lost at the output

Multi-head attention

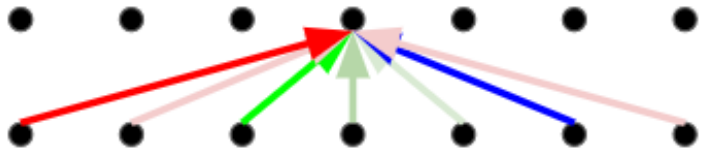
- Multiple attention layers (heads) that run in parallel
- Each head use different weights
- Each head can learn different relationship



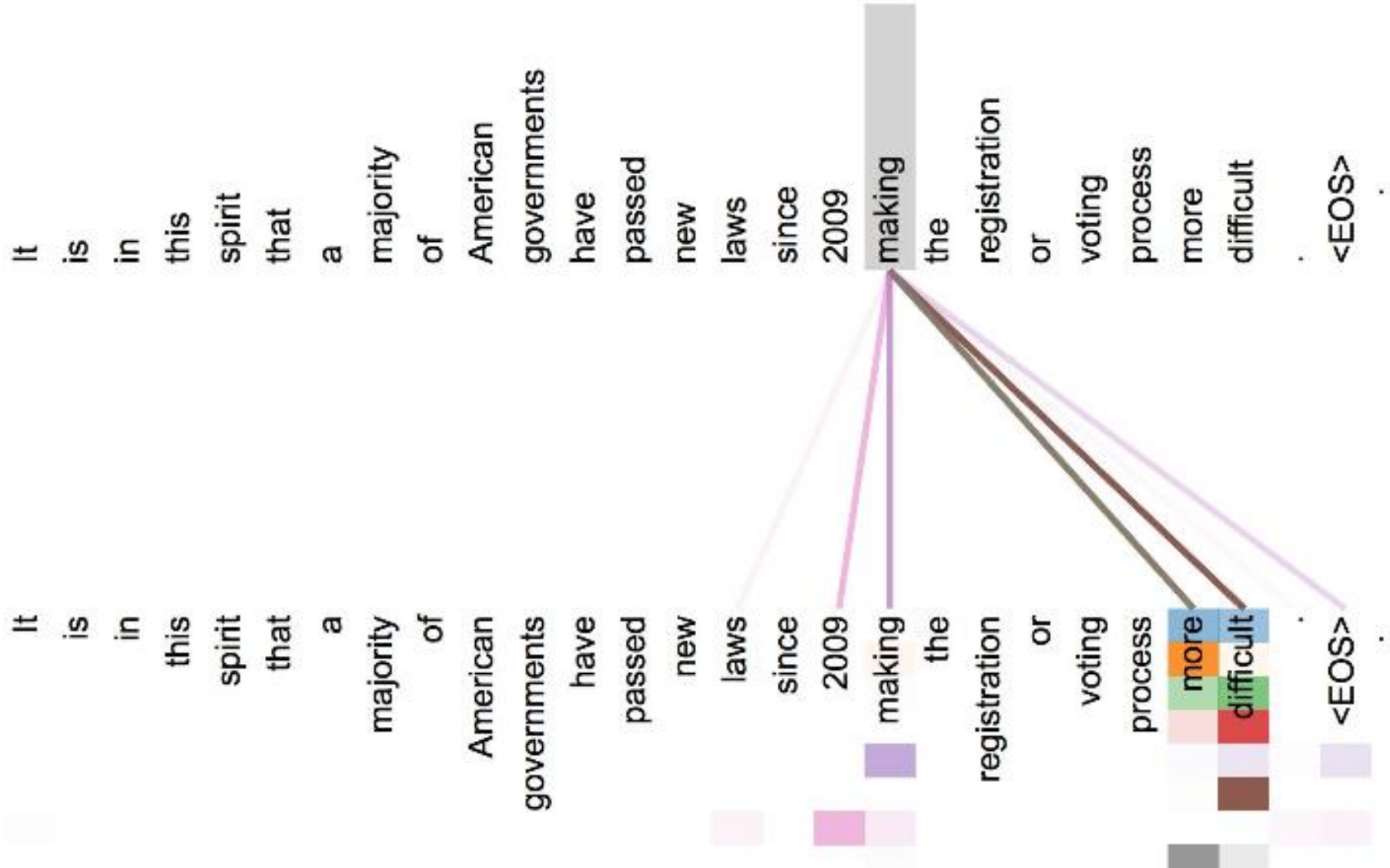
Convolution

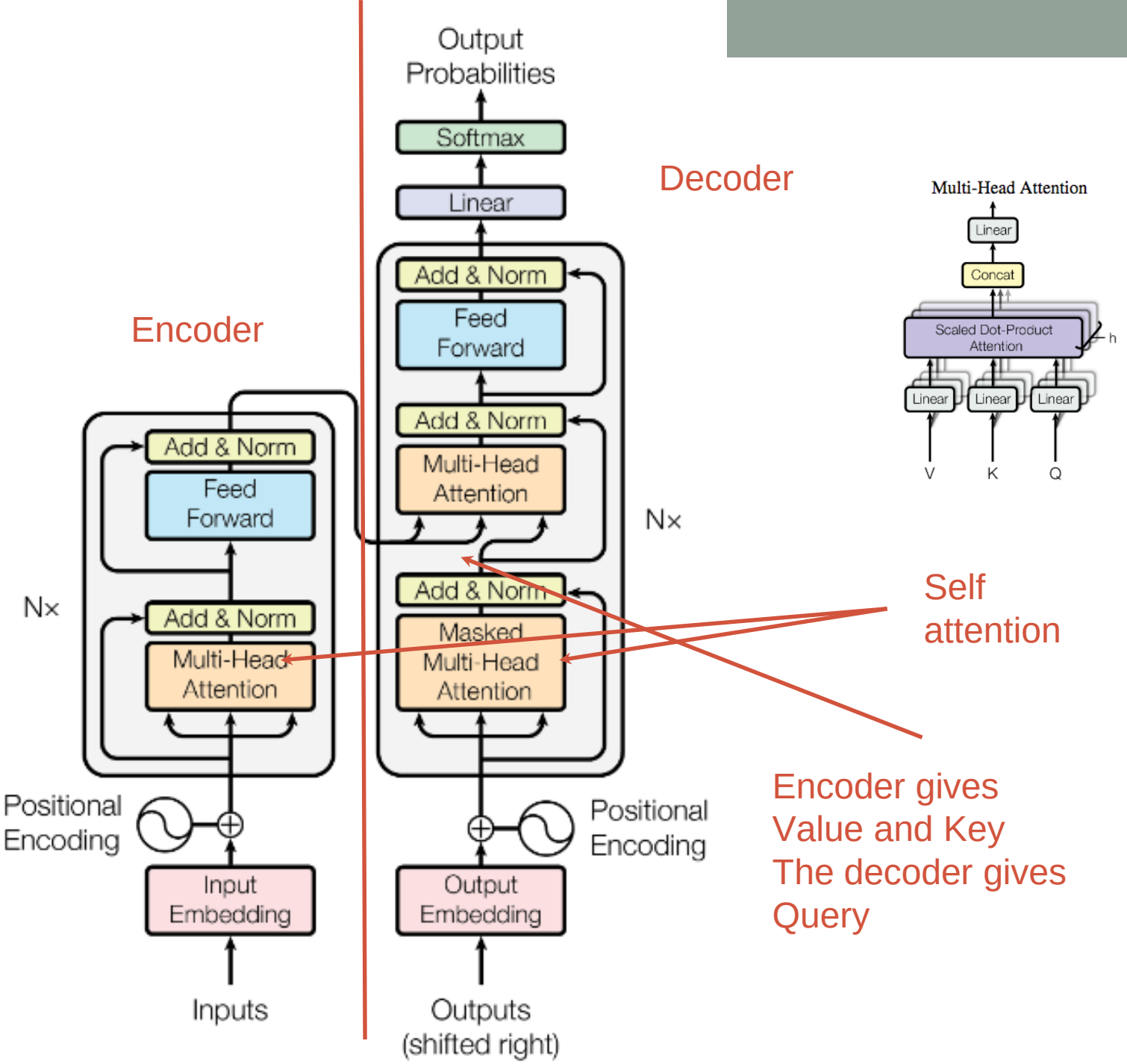


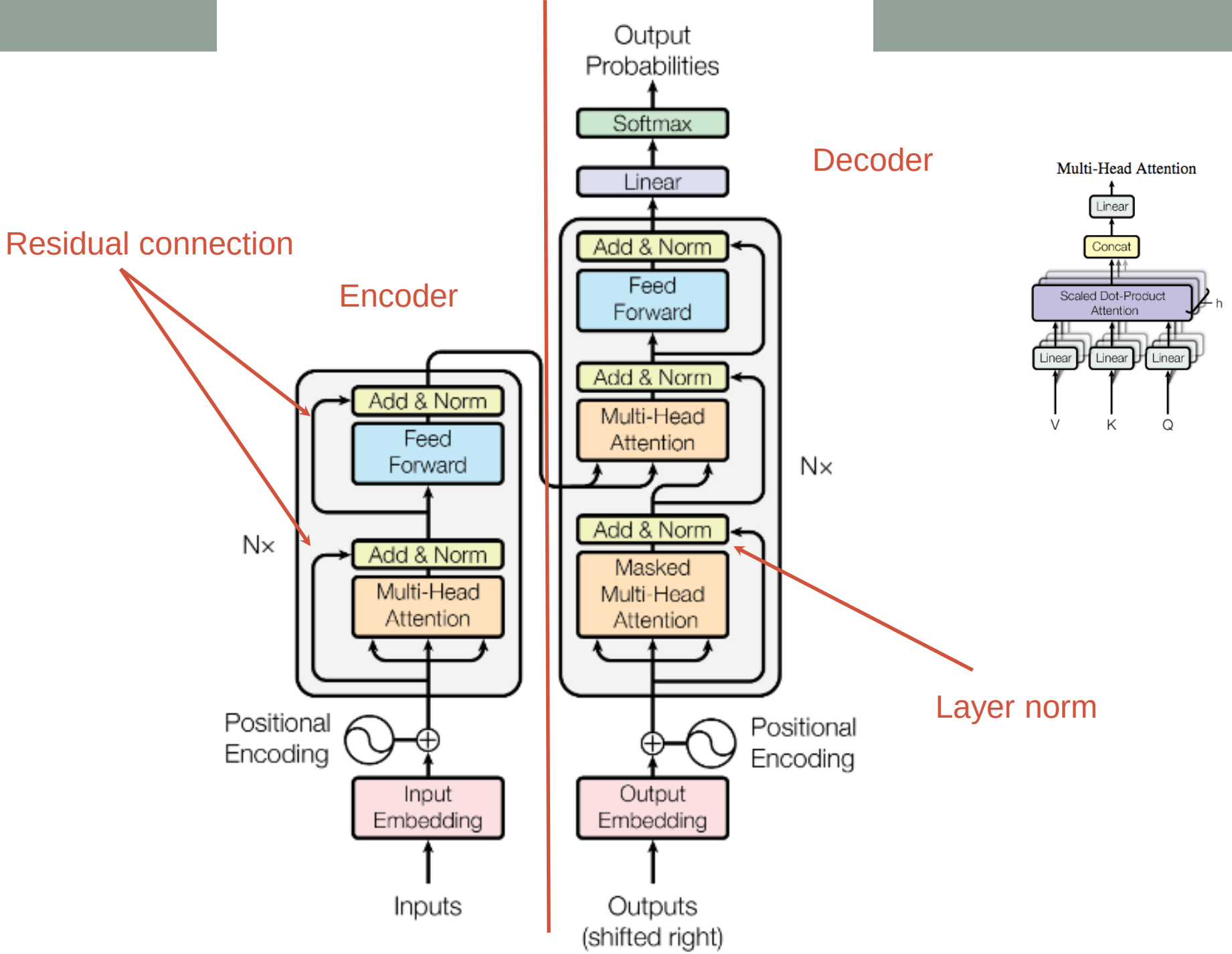
Multi-Head Attention



Multi-head visualization

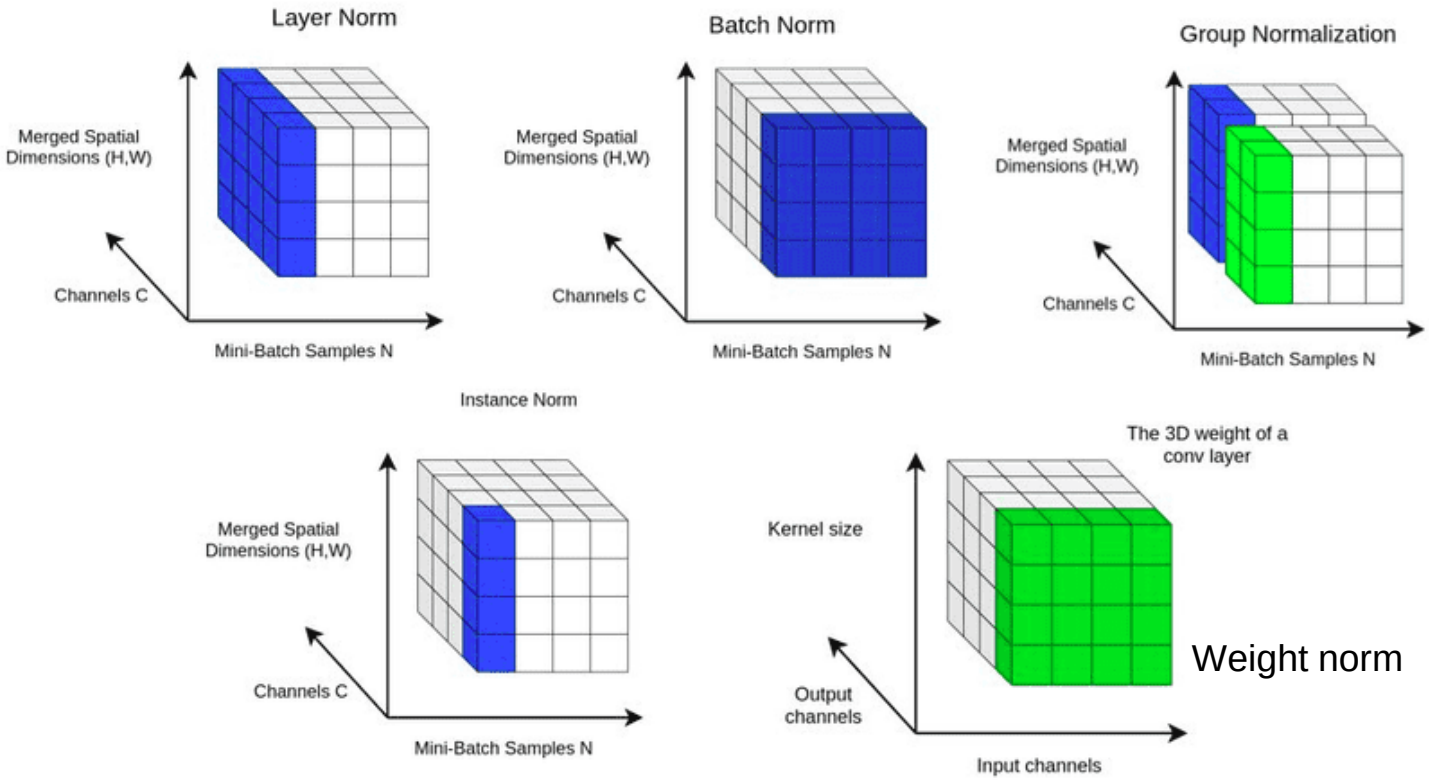



Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Nx
Nx
Encoder
Decoder
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
Self attention
Encoder gives Value and Key
The decoder gives Query

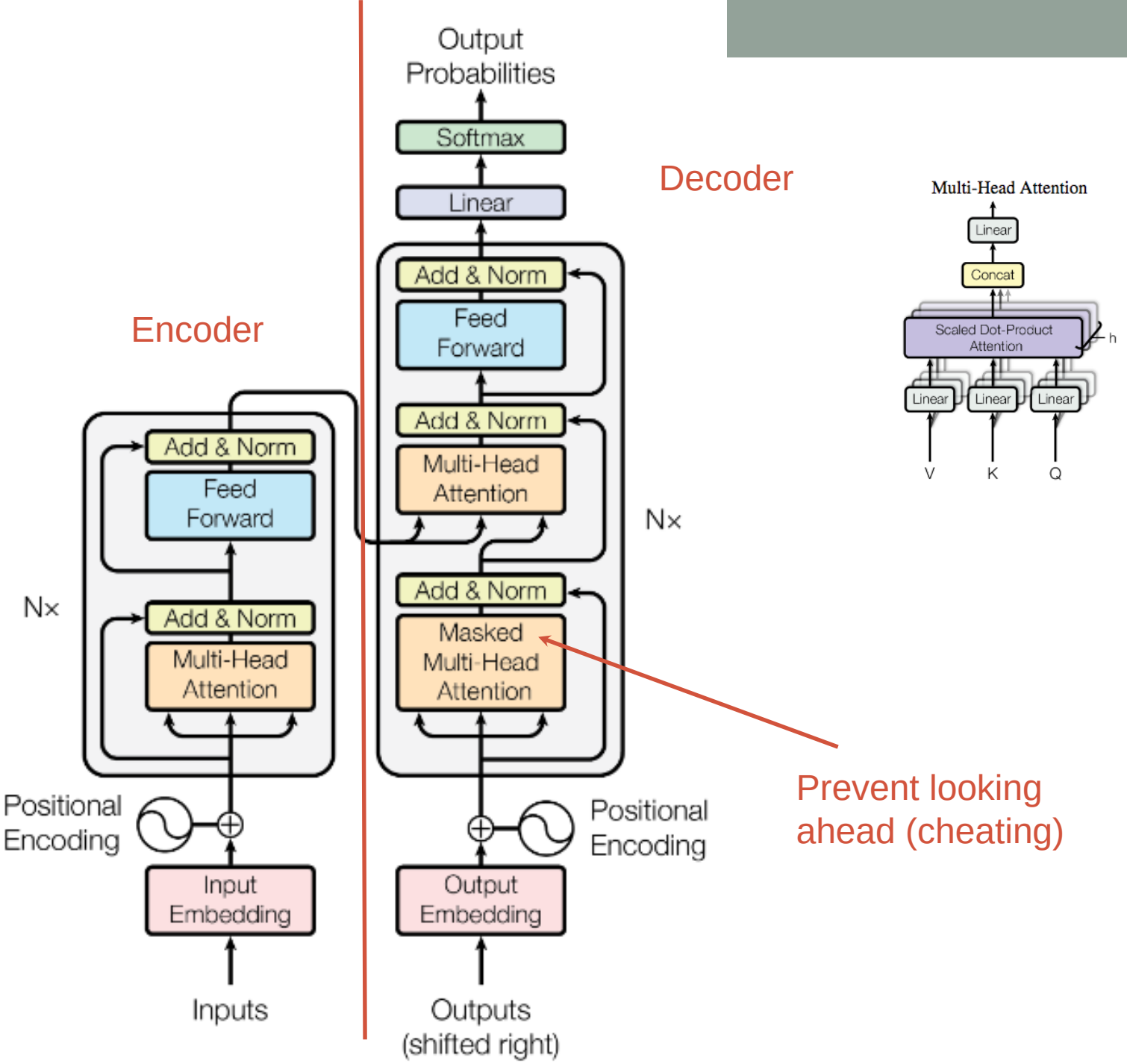

Residual connection
Encoder
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding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

Other normalizations

- Other normalizations are out there



https://theaisummer.com/normalization/


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

MT results

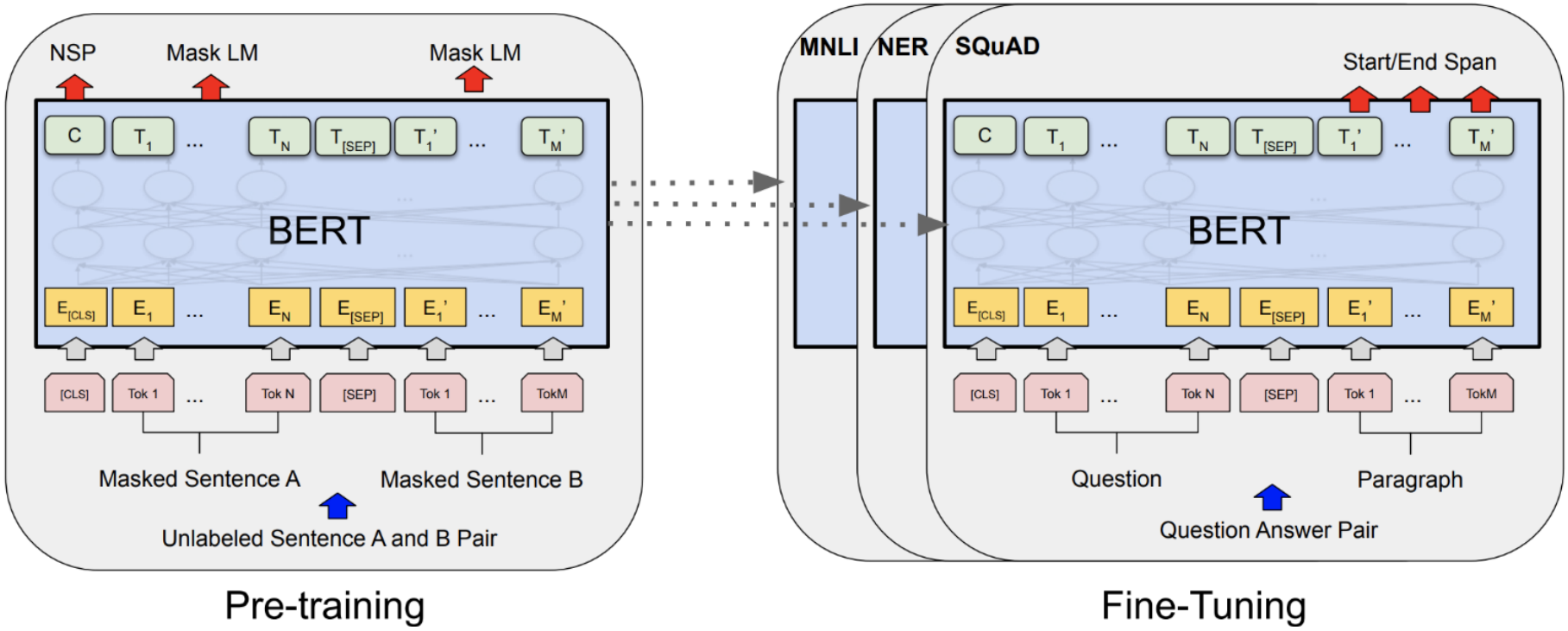
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

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.

BERT

Full transformer
Masked LM to learn bi-directional LM
Next sentence prediction to learn discourse



https://arxiv.org/abs/1810.04805

Pre-trained transformer types (by training method)

Encoder only (autoencoder)

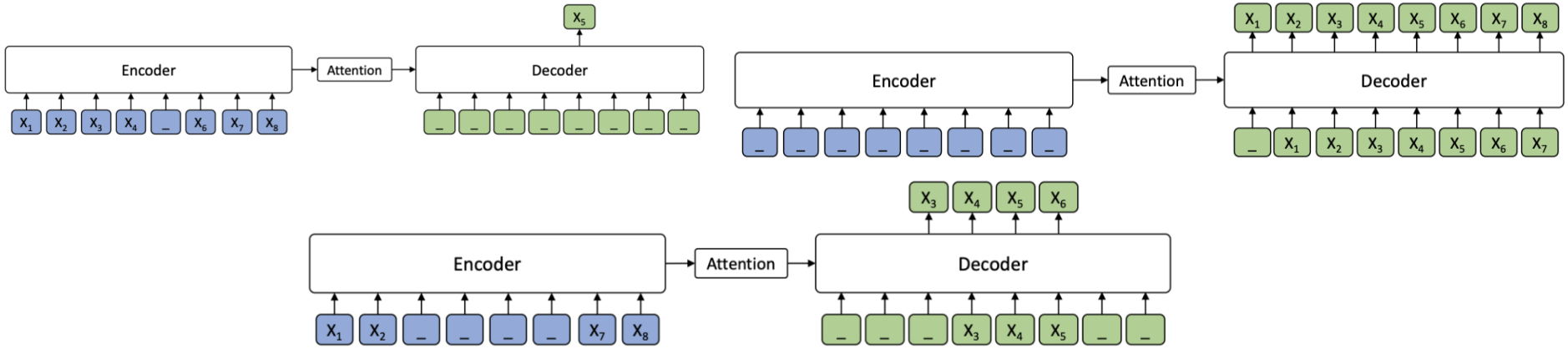
- BERT, ALBERT, RoBERTa
- Seq classification, token classification
- Masked words and predict

Encoder-decoder (seq2seq)

- MASS, BART, T5
- Machine translation, text summary
- Mask phrases

Decoder only (autoregressive)

- GPT, CTRL
- Text generation
- Predict next word

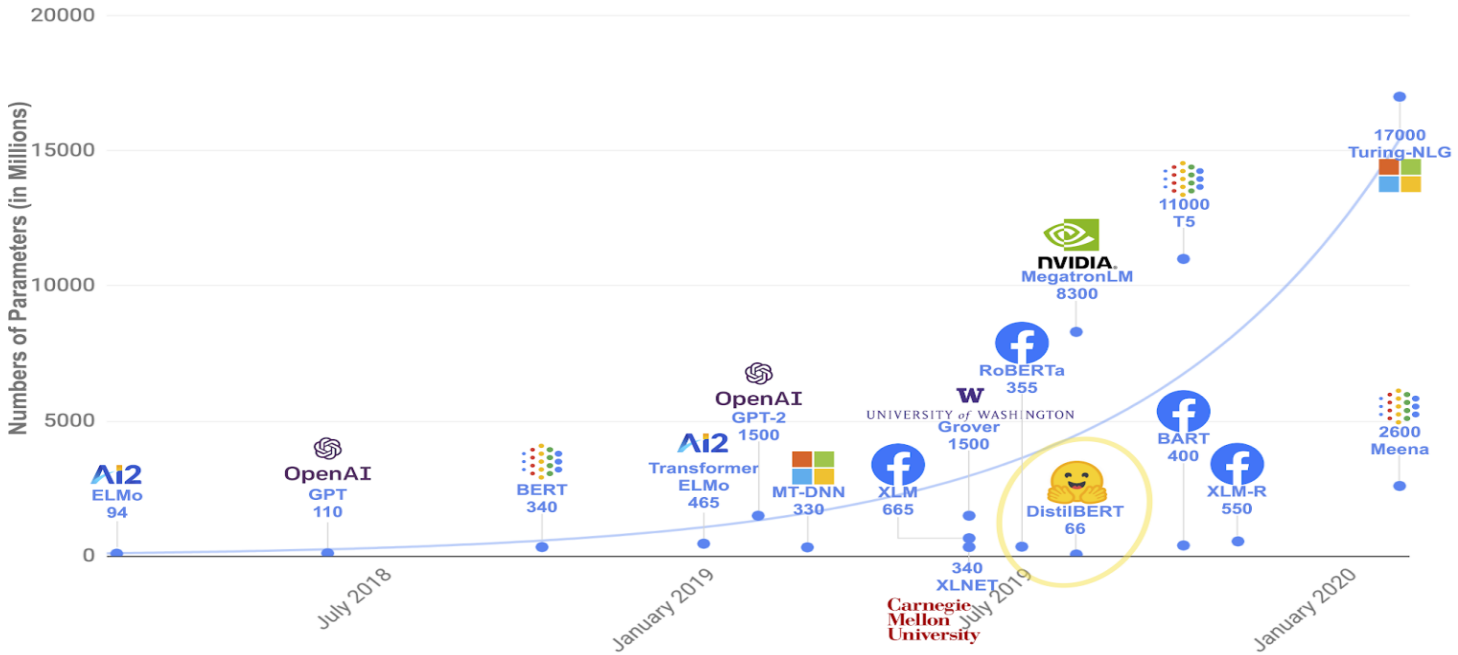


https://huggingface.co/transformers/summary.html https://arxiv.org/pdf/1905.02450.pdf

Recent trends in transfomers

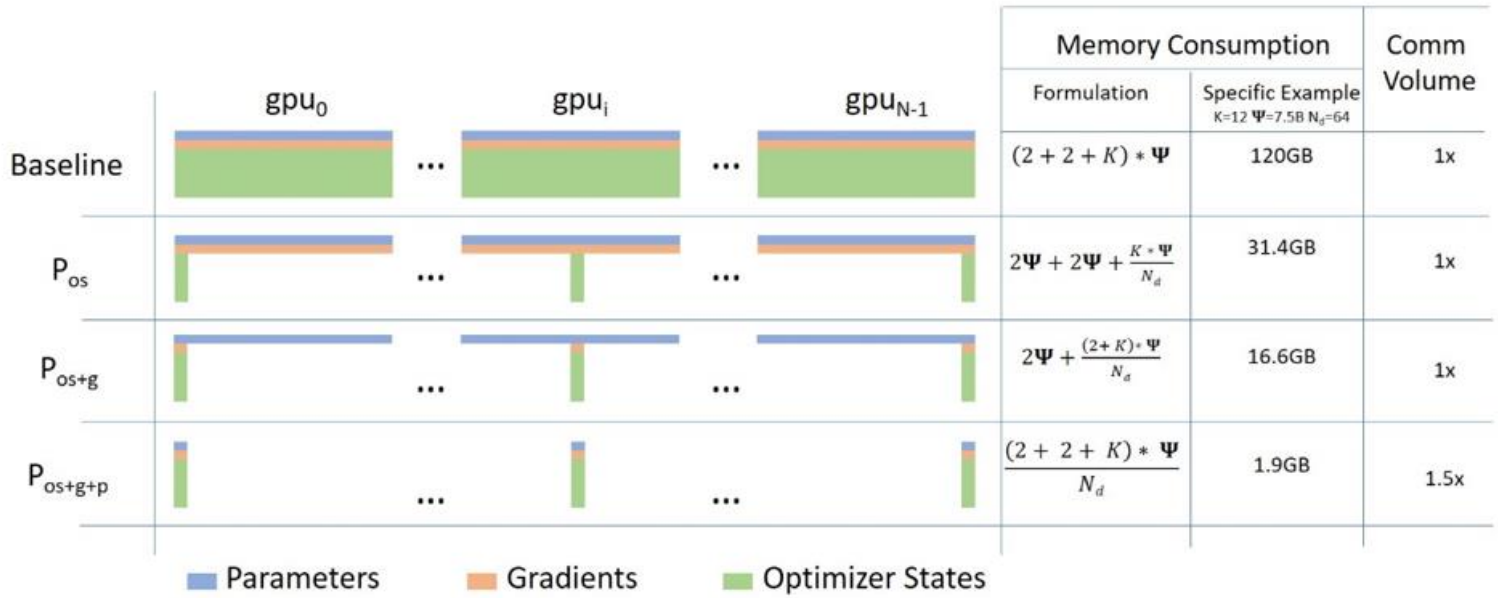
- Larger models

  Possible by more efficient architectures, more data, more compute, smarter distributed training



Zero & DeepSpeed

Clever management of training parameters across GPUs and machines



	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

T-NLG

	LAMBADA (acc) strict	WikiText-103 (test adj. ppl)
Open AI GPT-2 1.5B	52.66 (63.24)*	17.48
Megatron-LM 8.3B	66.51	10.81
T-NLG 17B	67.98	10.21

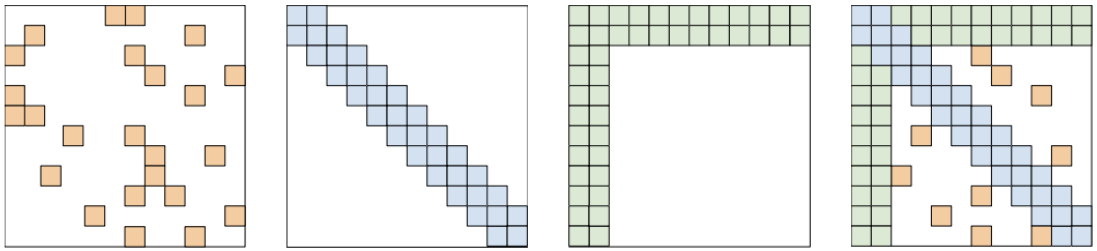
*Open AI used additional processing (stopword filtering) to achieve higher numbers than the model achieved alone. Neither Megatron nor T-NLG use this stopword filtering technique.

Can do Q/A by just LOTs of internet text

When did WW2 end?	WW2 ended in 1945.
How many people live in the US?	There are over 300 million people living in the US.

Big bird

- Sparse attention pattern to reduce computation
  - Random attention: inspired by random graph (strong connectivity)
  - Window attention: most words attend locally
  - Global attention: sentence tokens should attend globally



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

Tricks to make better transformers

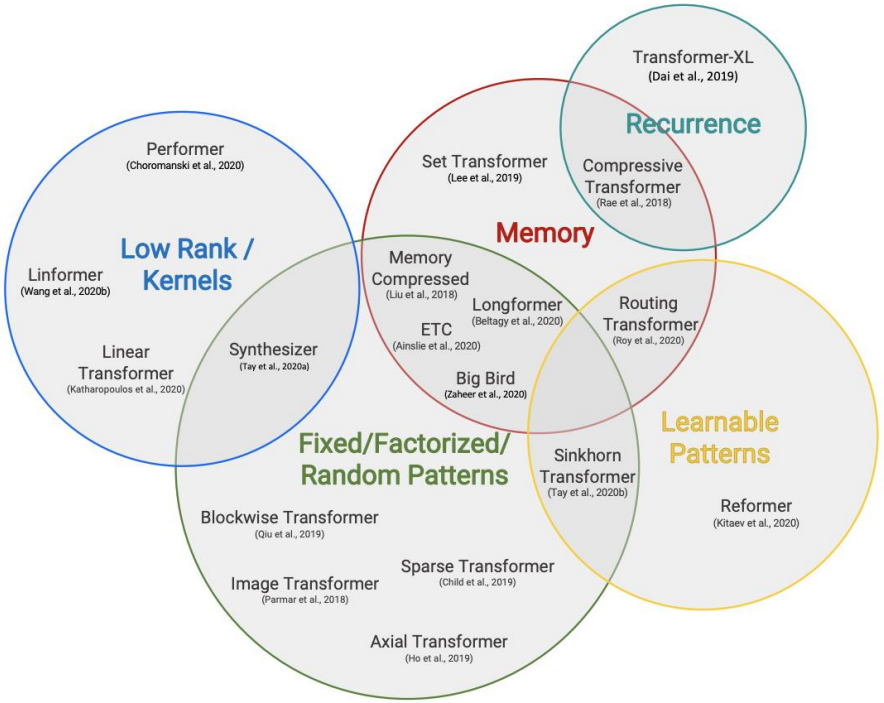
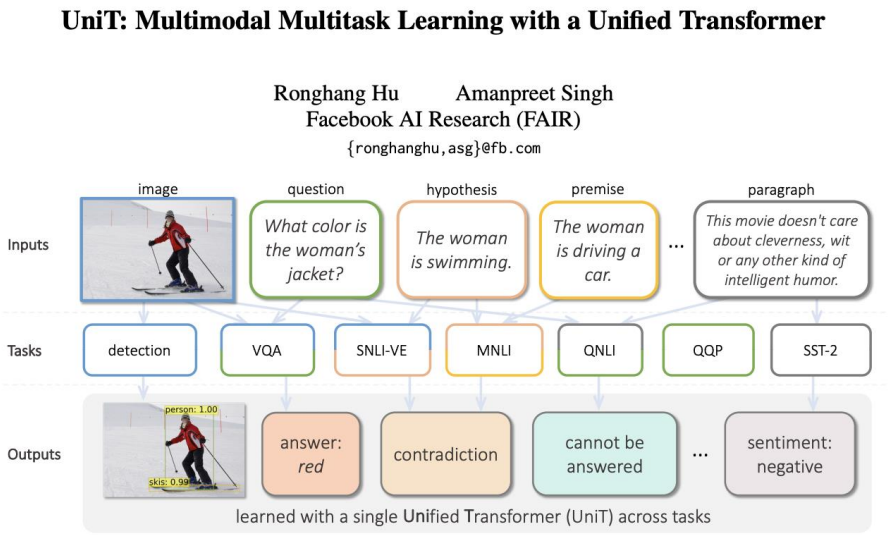


Figure 2: Taxonomy of Efficient Transformer Architectures.

Efficient Transformers: A Survey, 2020 https://arxiv.org/pdf/2009.06732.pdf

Trends

- Transformer is expensive
- Transformer replaces CNN-based architecture
- Transformer is all you need



UniT: Multimodal Multitask Learning with a Unified Transformer, 2021

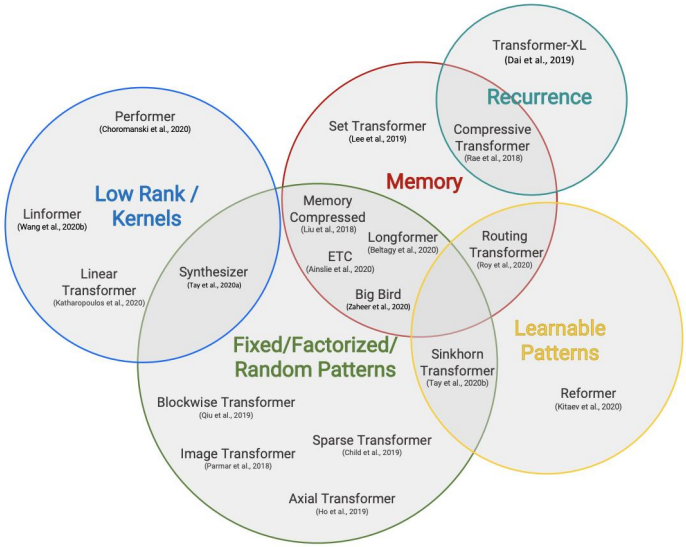


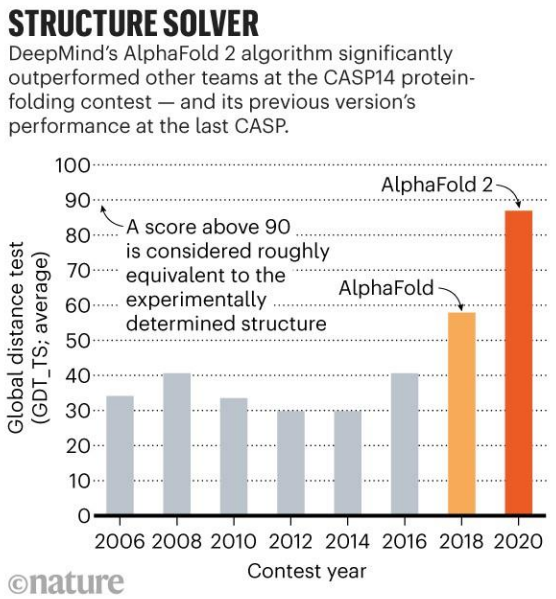
Figure 2: Taxonomy of Efficient Transformer Architectures.

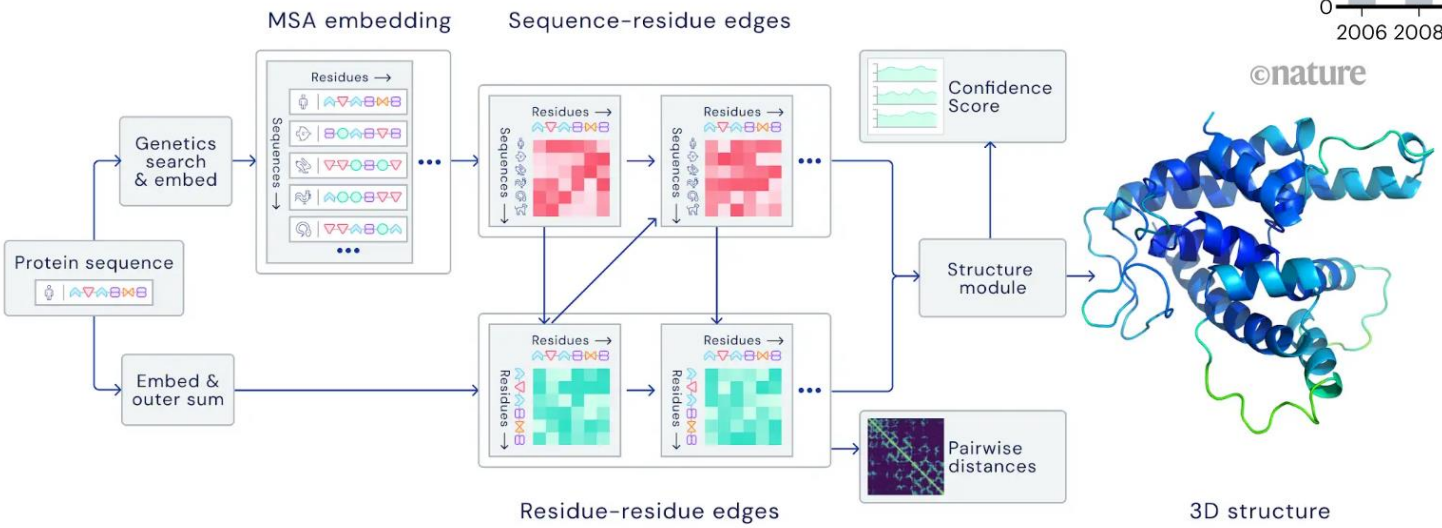
Efficient Transformers: A Survey, 2020

Alphafold2

https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

https://www.nature.com/articles/d41586-020-03348-4





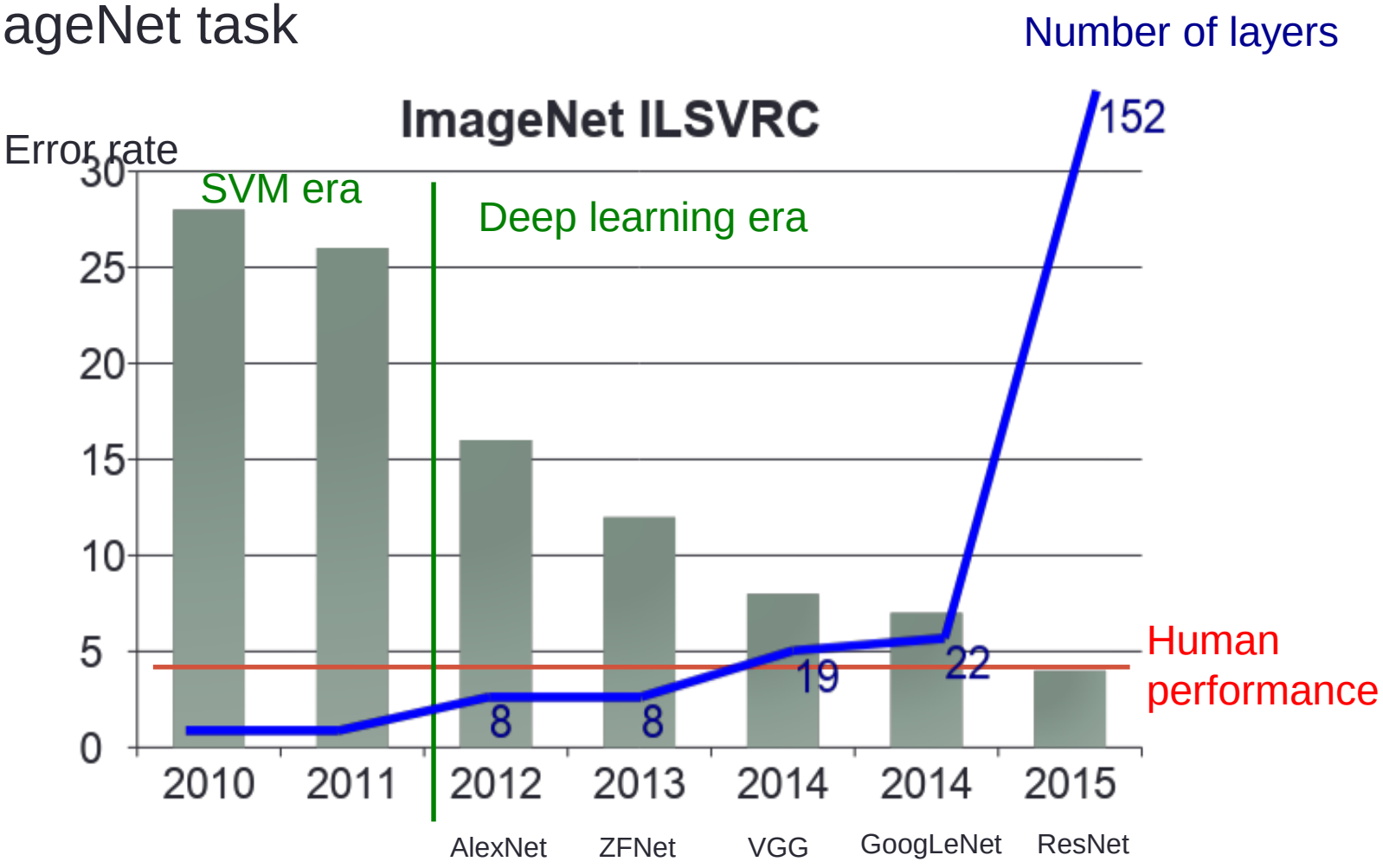
Structure of this lecture

- Attention
- Transformer
- Transformer is all you need?

Back to vision world

A brief history of imagenet architectures

- ImageNet task



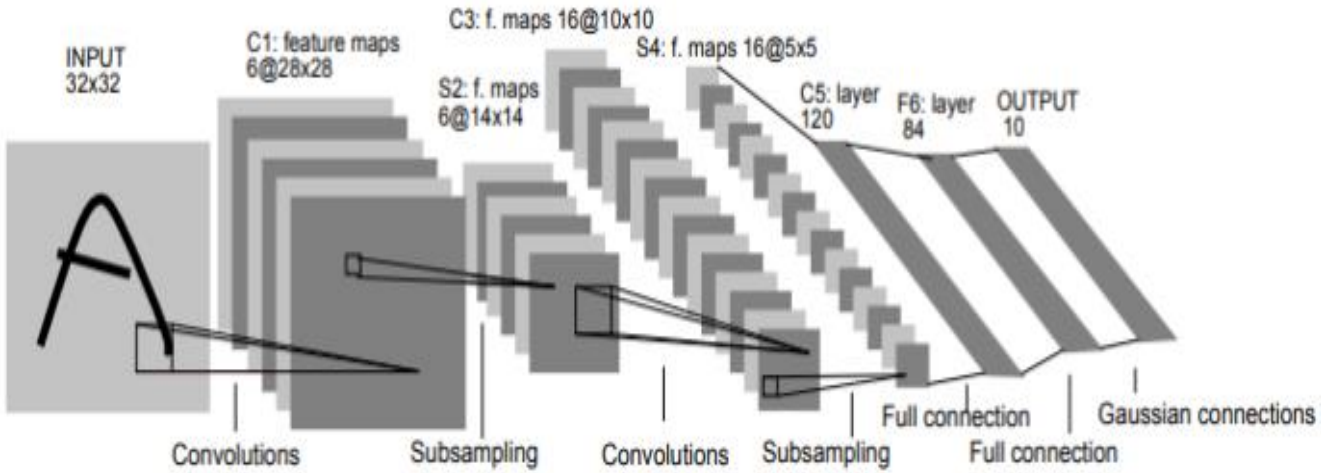
Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

LeNet

Convolutions and poolings followed by fully connected layers

Tanh activations

Ability to handle larger images limited by compute

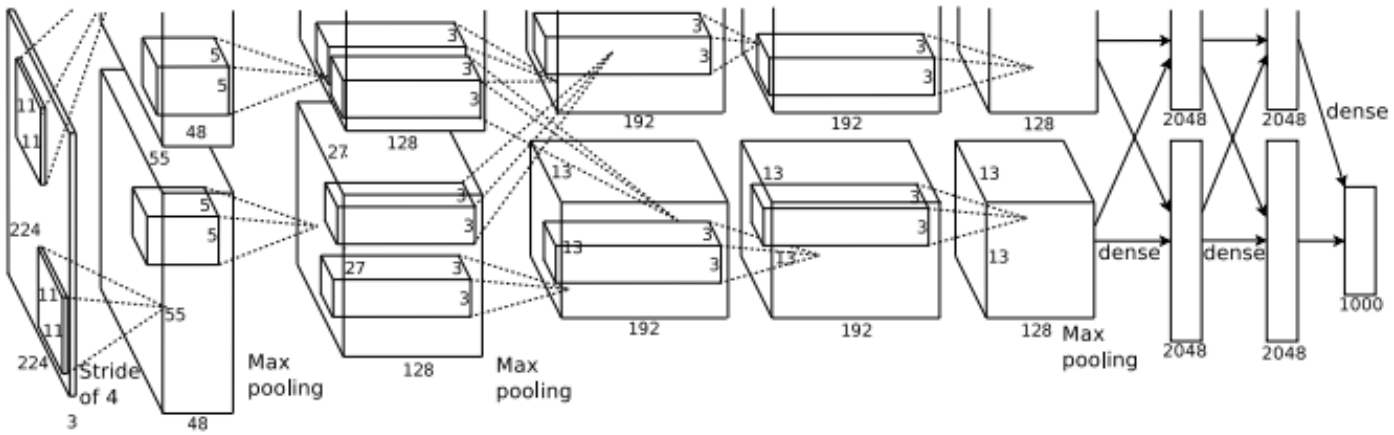


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition,” 1998.

AlexNet

Convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum

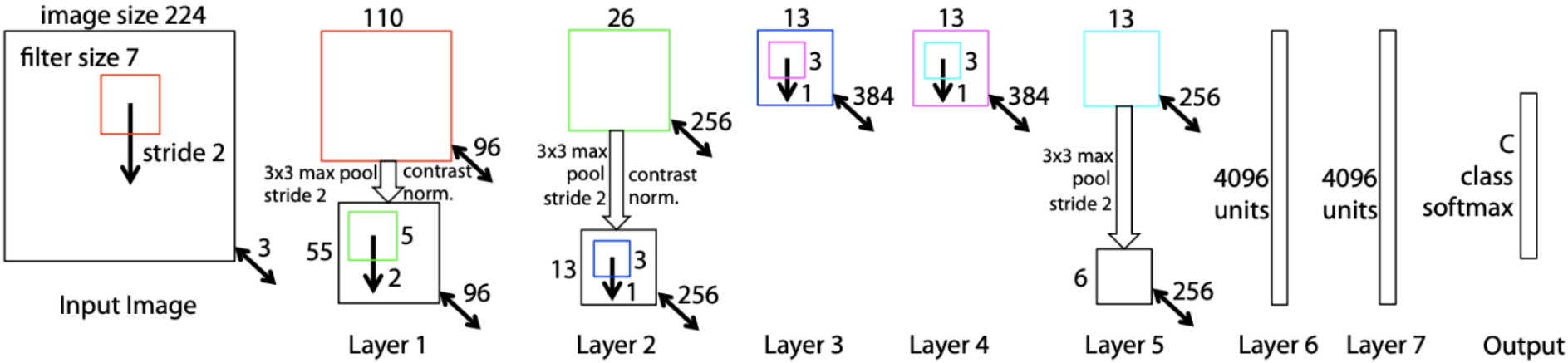
Two pipelines to fit into two GPUs



Alex Krizhevsky, et al. “ImageNet Classification with Deep Convolutional Neural Networks”, 2012.

ZFNet

Tweaking hyperparameters



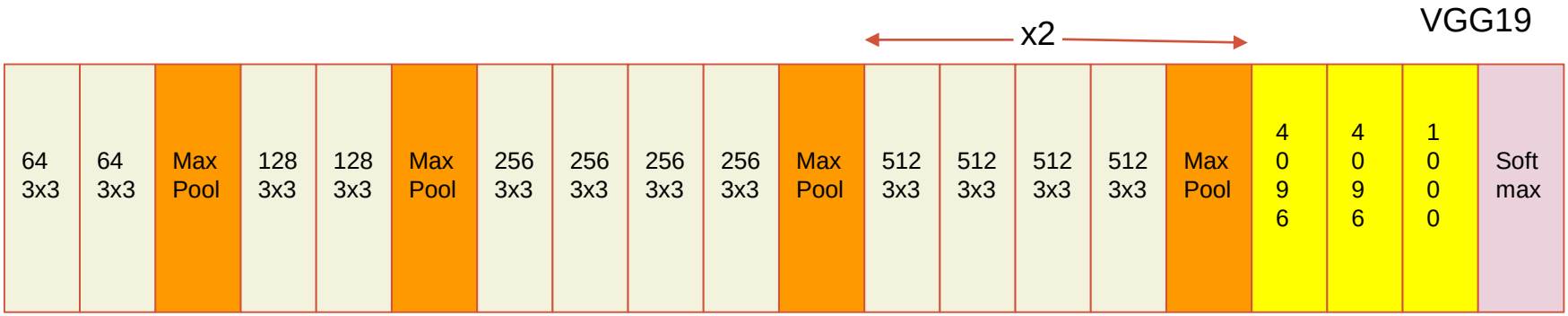
Matthew D. Zeiler, Rob Fergus, “Visualizing and Understanding Convolutional Networks,” 2013

VGG

Uniform 3x3 convolutional filters

19 layers! Pushing the limits of conventional wisdom at that time.

Used by many since pre-train weights are publically available



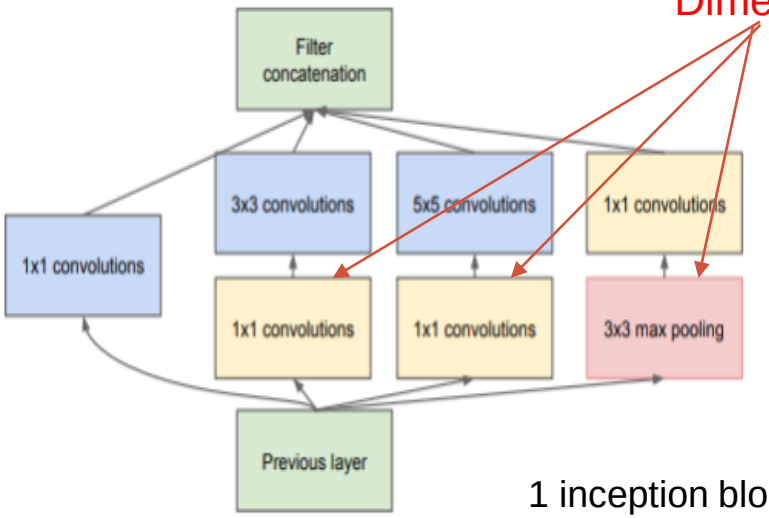
Karen Simonyan, Andrew Zisserman"Very Deep Convolutional Networks for Large-Scale Image Recognition" 2014

GoogLeNet (Inception v1)

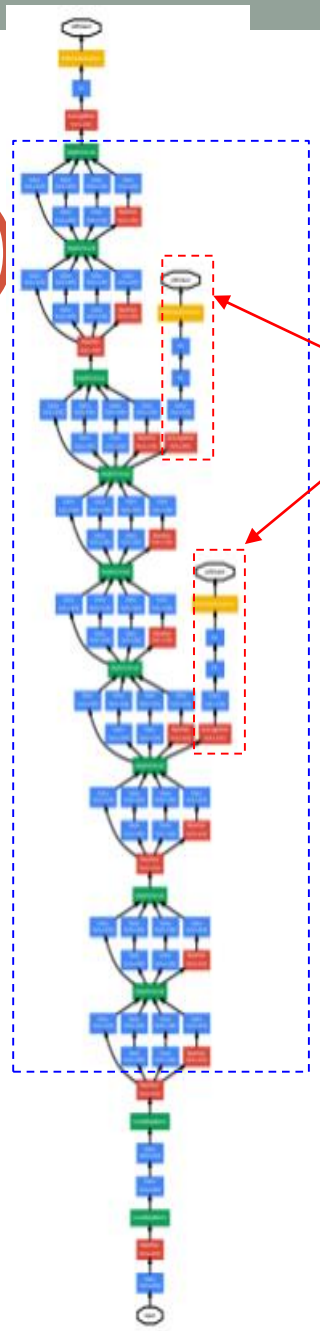
Multiple filter sizes per layer (objects come in different scales)

Dimensionality reduction via 1x1 convolution

Multiple softmax losses to help the gradient problem



1 inception block

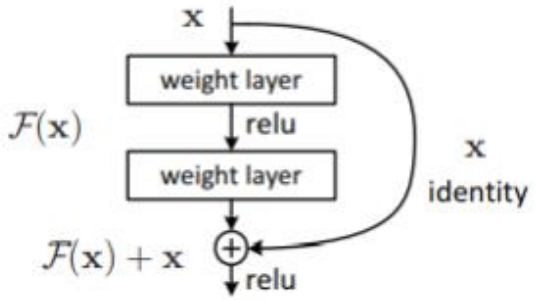


Christian Szegedy, et al. “Going Deeper with Convolutions,” 2014

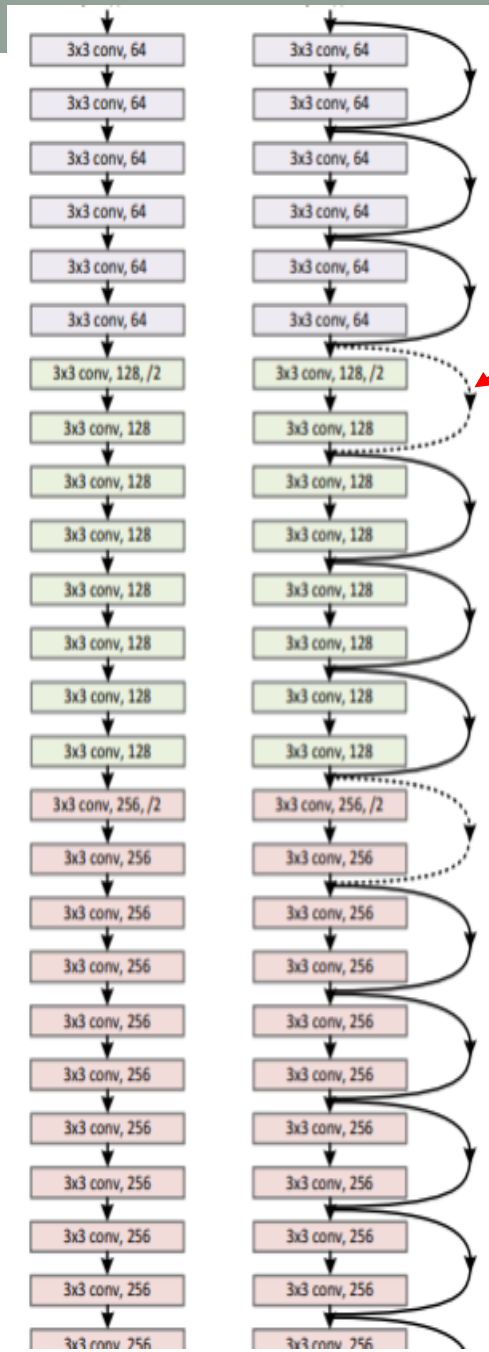
ResNet (Residual Network)

Batch norm

Extra “skip connections” to reduce the vanishing gradient problem



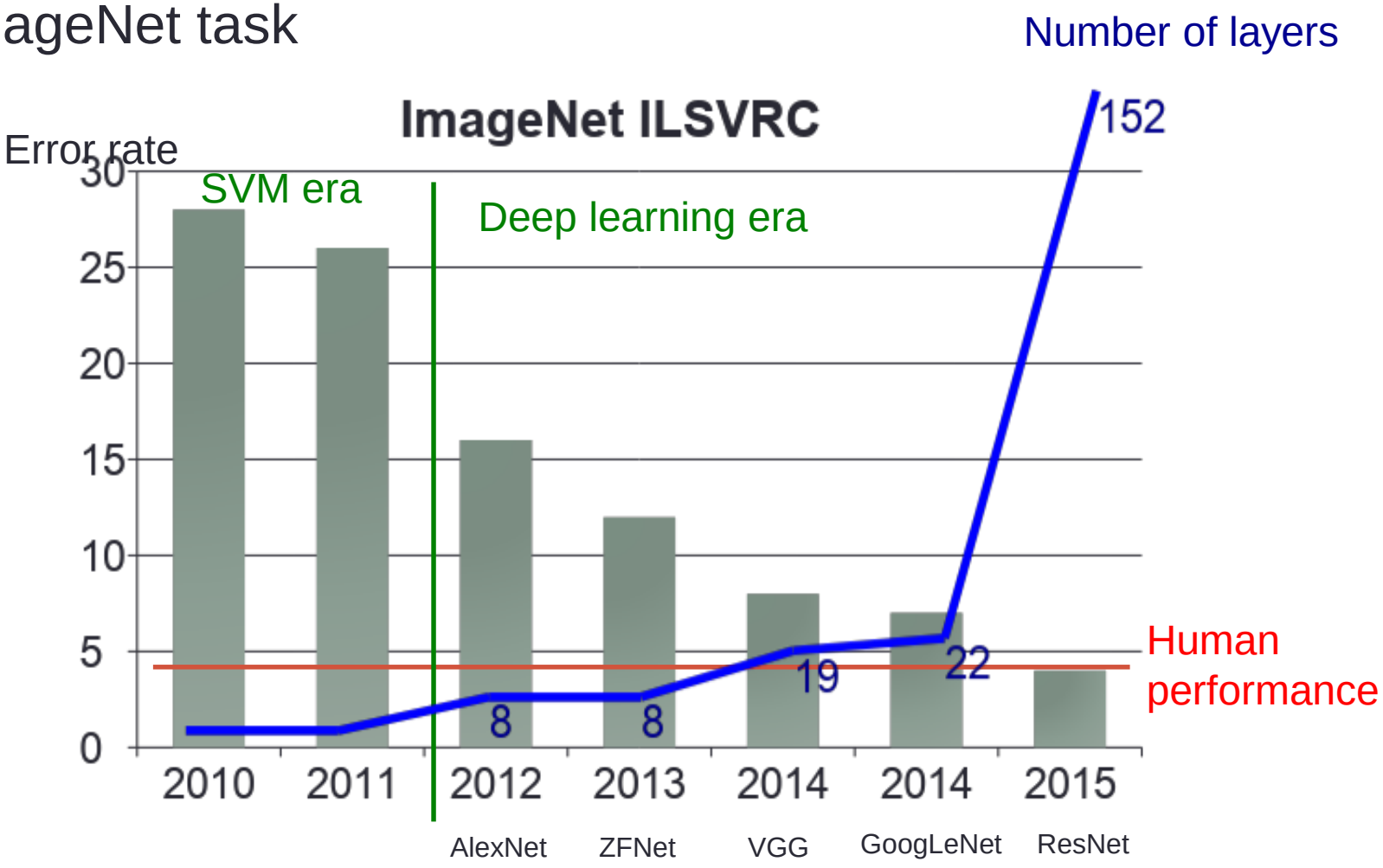
Kaiming He, et al. “Deep Residual Learning for Image Recognition” 2015



Need extra operations to make the size the same

A brief history of imagenet architectures

- ImageNet task



Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

Inception v2+v3

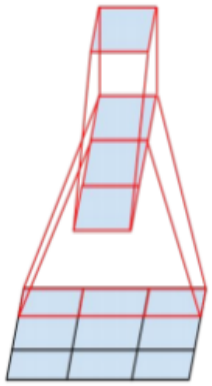
Implement 5x5 with two 3x3s

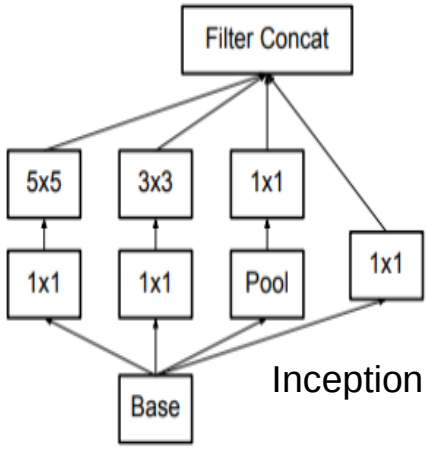
Factorized convolution

3x3 -> 3x1 and 1x3

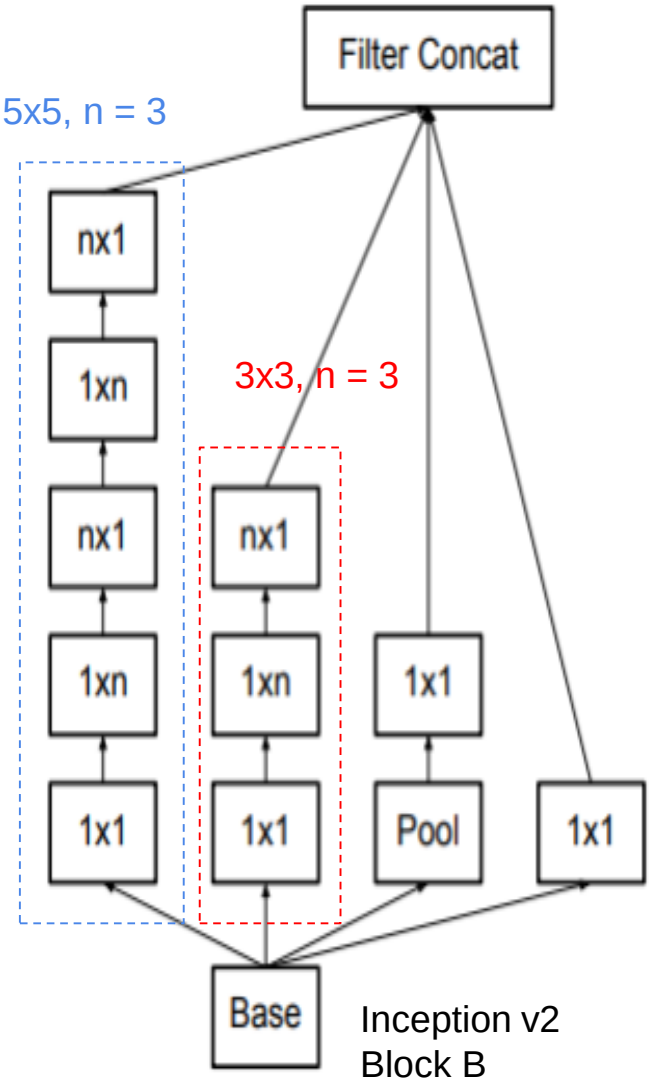
3 types of inception blocks

RMSprop, Batch norm, label smoothing





Inception v1



Inception v2
Block B

Christian Szegedy, “Rethinking the Inception Architecture for Computer Vision,” 2015

Xception

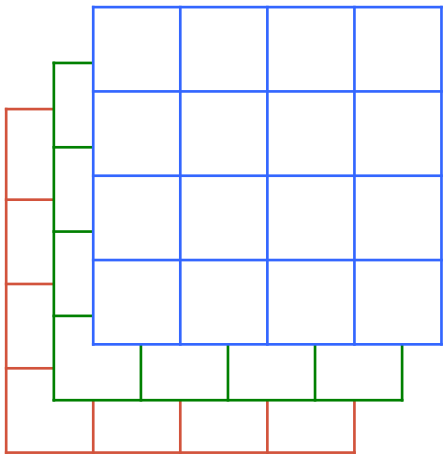
Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution

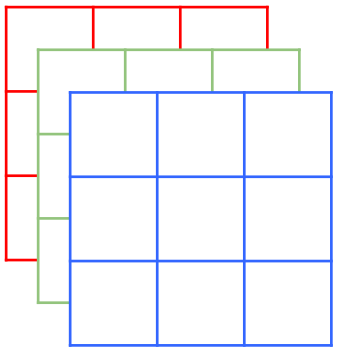
Typical convolution 3x3 filter is 3x3xinput channel

Depthwise convolution 3x3 filter is 3x3x1
1 filter per input channel

input



One regular convolution filter



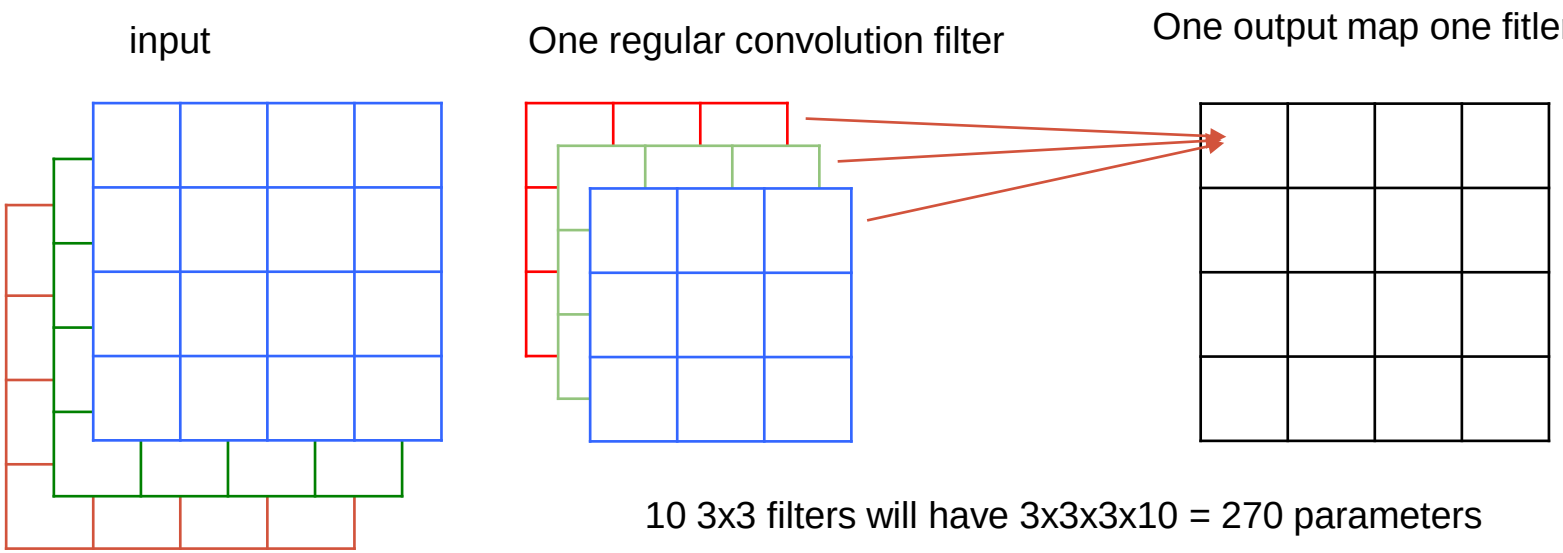
Xception

Depthwise separable convolution: two-step convolution

1. Depthwise convolution
2. 1x1 convolution

Typical convolution 3x3 filter is 3x3xinput channel

Depthwise convolution 3x3 filter is 3x3x1
1 filter per input channel

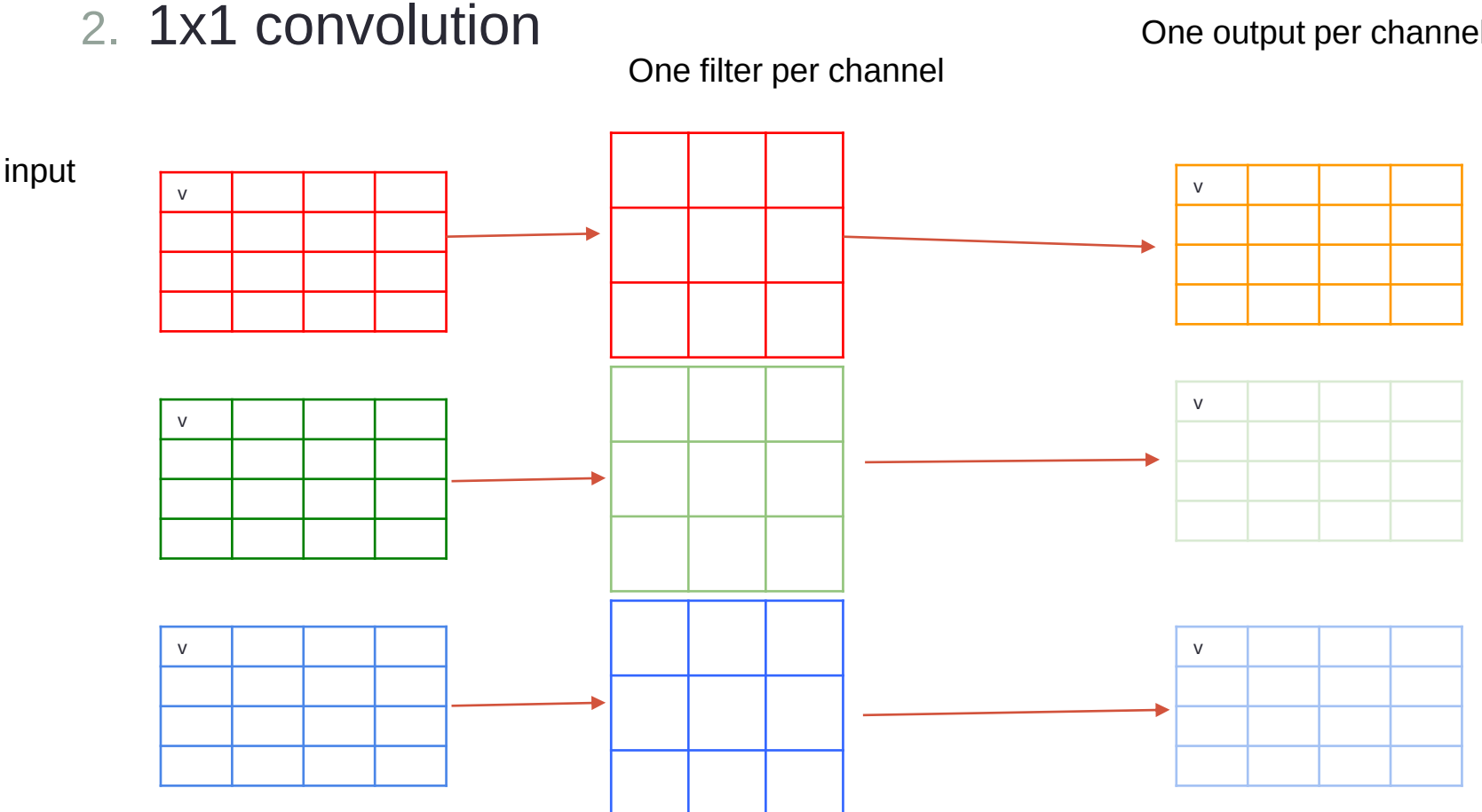


10 3x3 filters will have 3x3x3x10 = 270 parameters

Xception

Depthwise separable convolution: two-step convolution

1. Depthwise convolution (learn from each channel, spatial info)
2. 1x1 convolution



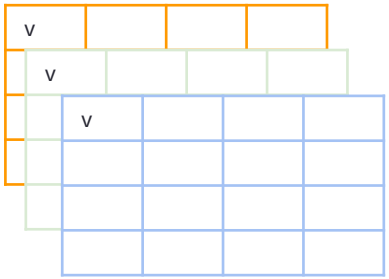
Xception

Depthwise separable convolution: two-step convolution

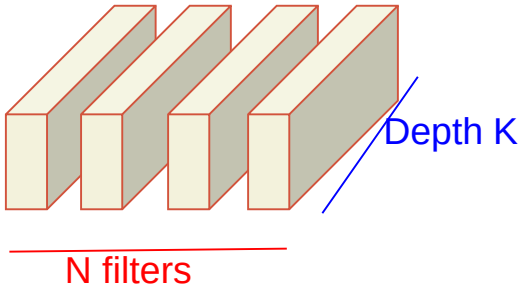
1. Depthwise convolution
2. 1x1 convolution (combine each channel, feature info)

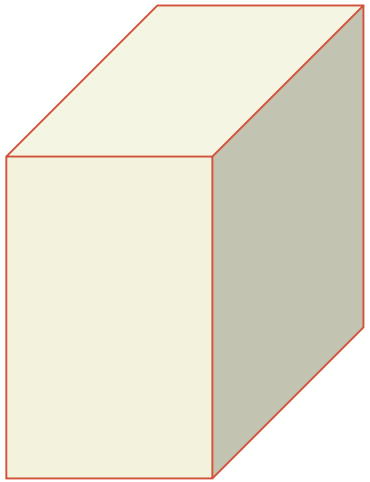
Final output

Output from depthwise convolution



1x1 filters





10 3x3 depthwise separable convoltuion will have 3x3x3 + 3x10 = 57 parameters

Xception

Replace convolutions in inception with depthwise separable convolutions

Smaller model

Faster compute

Comparable with Inception v3 while much faster

Used in MobileNet and other models

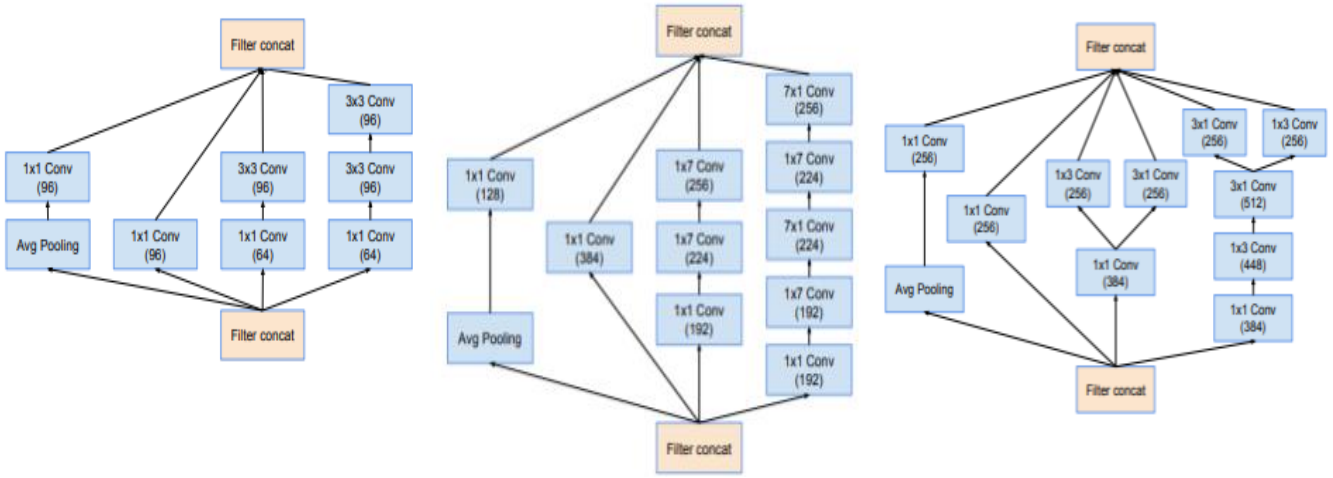
François Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions” 2016.
Andrew G. Howard, et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications” 2017.

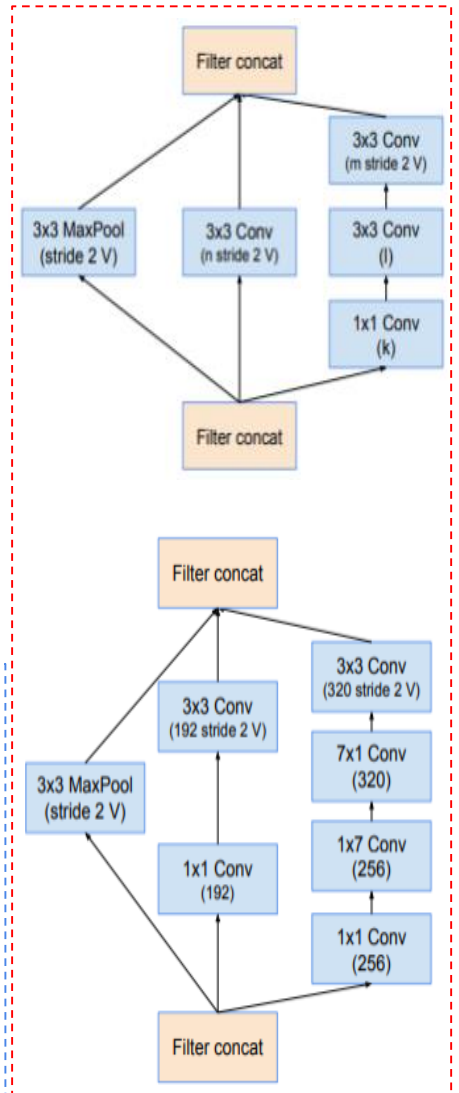
Inception v4

Same three types of inception blocks from v3

Add two types of reduction blocks for reducing the size of the grid (super pooling blocks)

Inception blocks





Reduction blocks

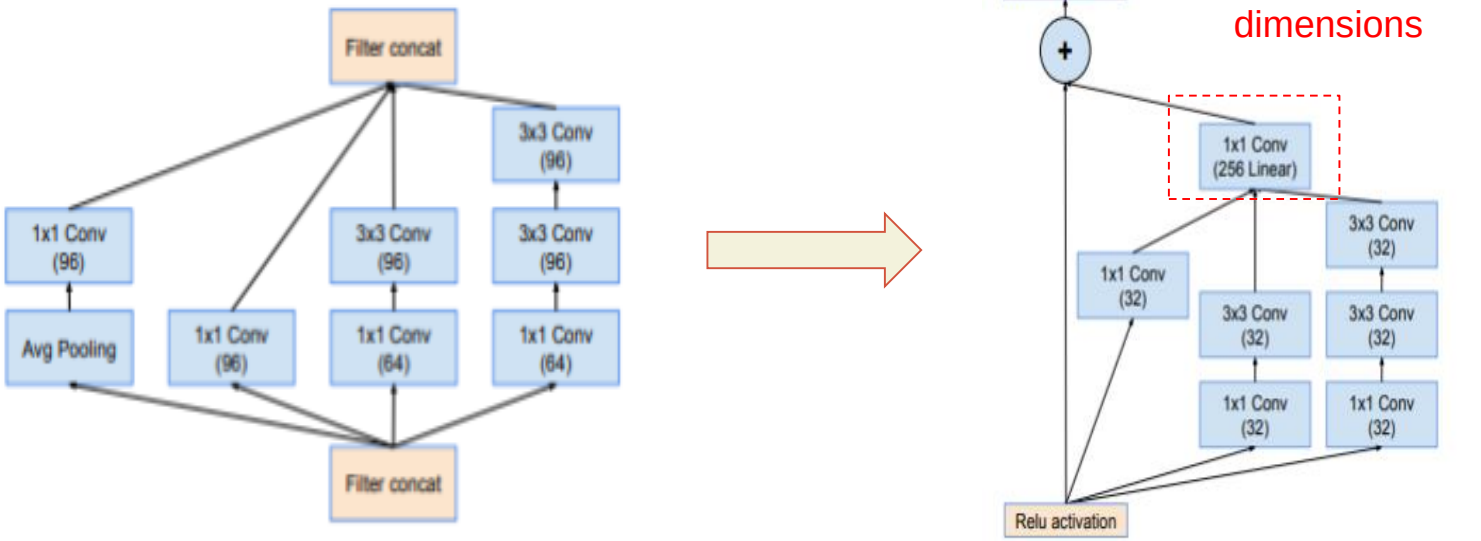
Christian Szegedy, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” 2016.

Inception-ResNet v1-v2

Introduce residual connections into inception blocks

Poolings changed to additions, 1x1 added to keep dimensions for the residual

Similar idea to ResNeXt



Christian Szegedy, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” 2016.

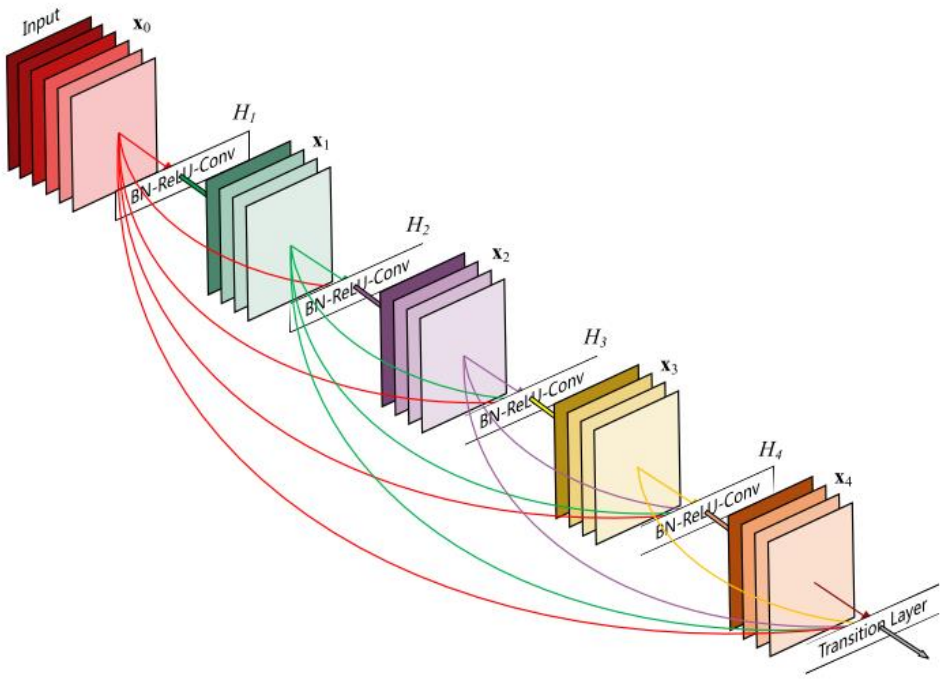
DenseNet

Instead of adding the residuals, concatenates the feature maps from previous layers

Densely connect multiple layers

Multi-scale feature maps

Smaller model due to smaller number of channels



Densely Connected Convolutional Networks 2016

Feature Pyramid Networks

- Not really a network but more like something on top of a network
- Multi-scale representation
  - Higher layer has large receptive field but low res
    - Good for large objects
  - Lower layer has small receptive field but high res
    - Good for small objects

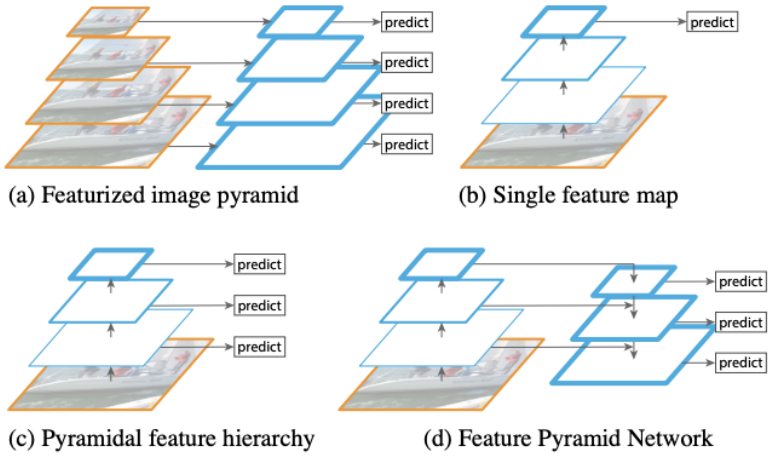
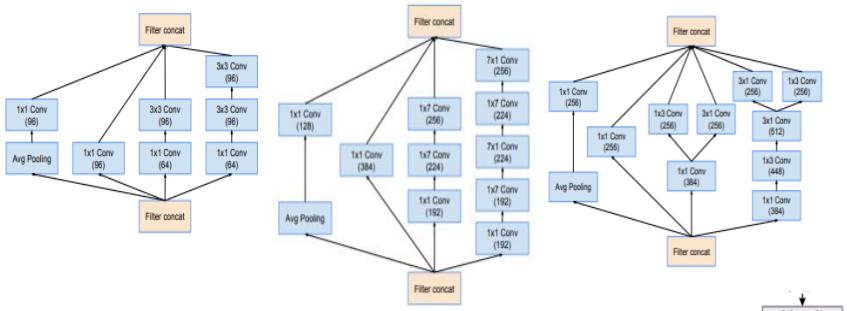
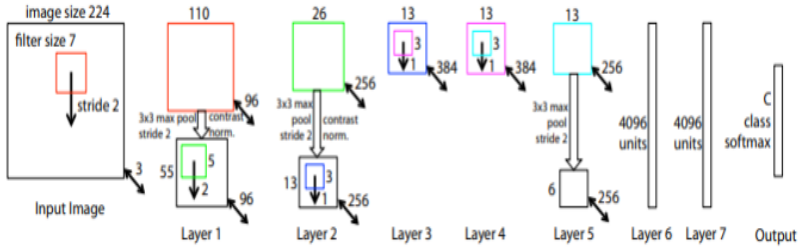


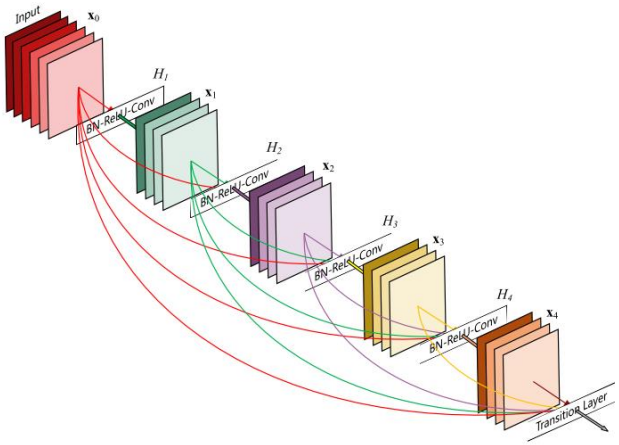
Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

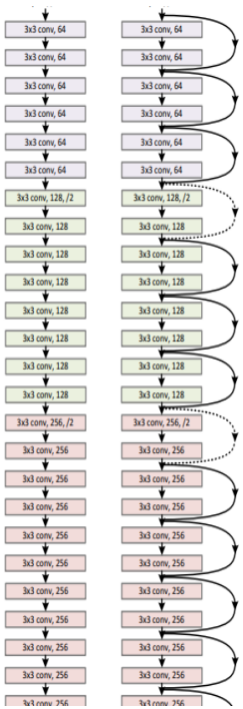
https://arxiv.org/abs/1612.03144

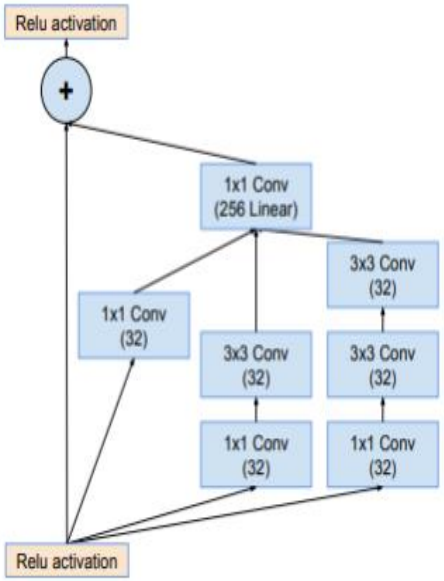
Do people keep tweaking network architectures until the end of time?









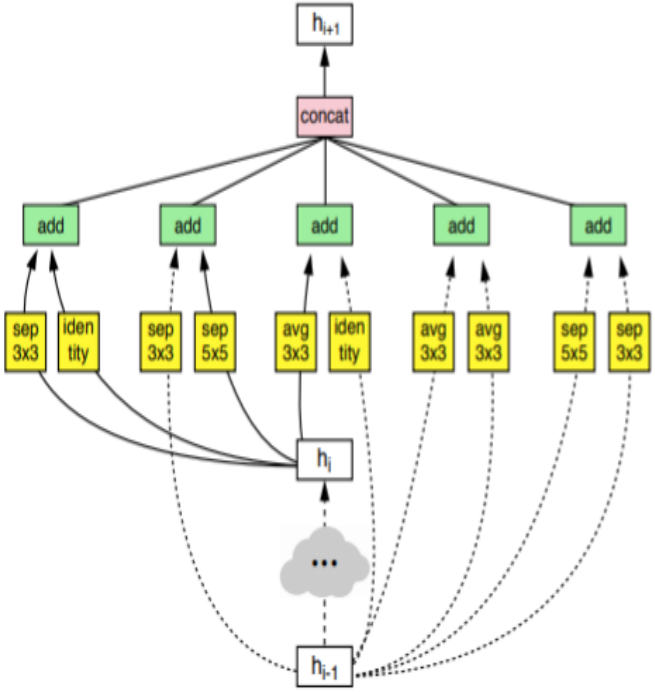


NasNet

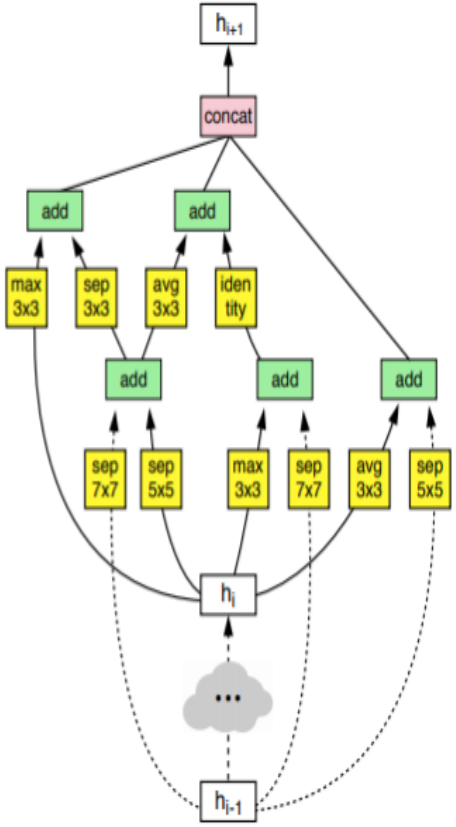
Neural Architecture Search Network

Use reinforcement learning to search for the best network configuration

Lowers compute (FLOPs) while maintaining high accuracy



Inception block



Reduction block

Barret Zoph, et al. “Learning Transferable Architectures for Scalable Image Recognition,” 2017.

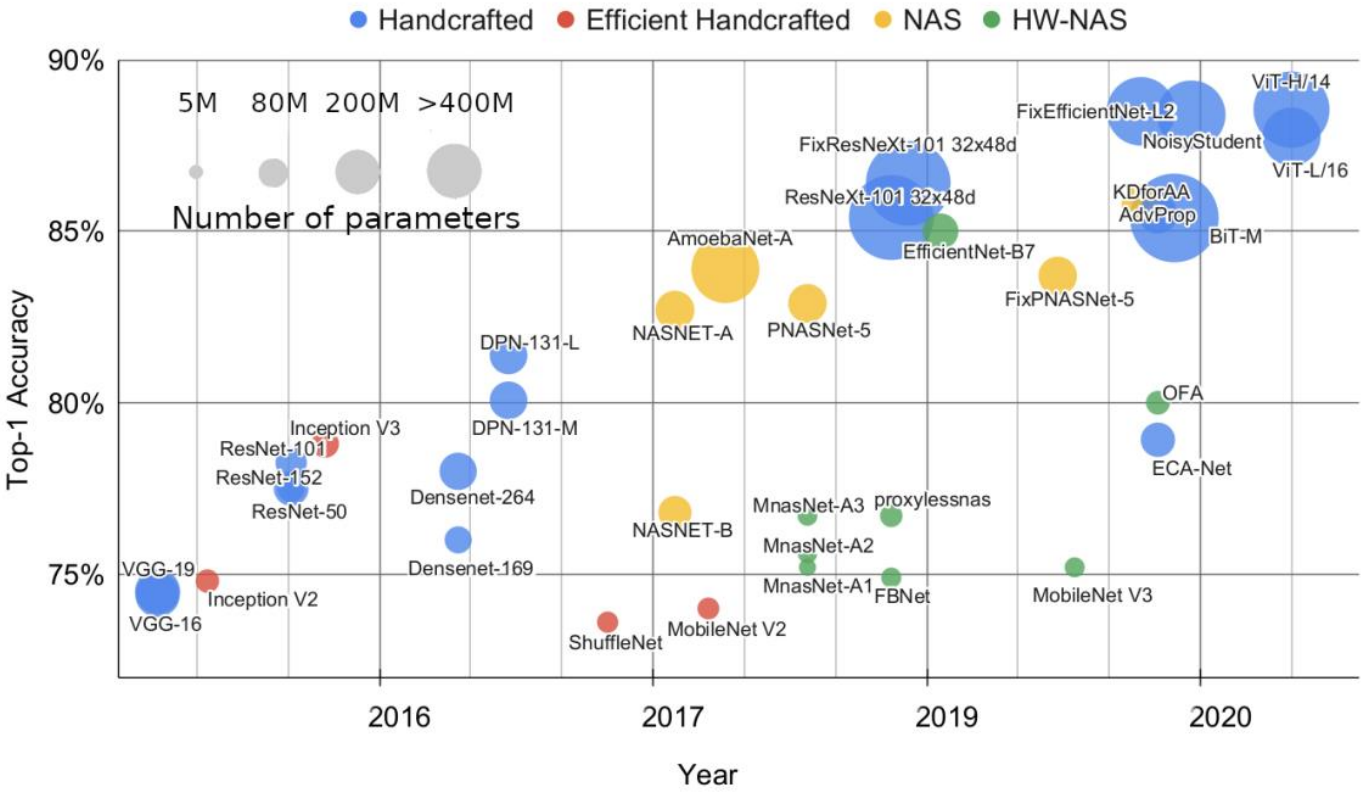
NAS doesn't know com sys arch

- ★ Response Time
  - Elapsed Time - Wall clock
  - CPU Time - Only CPU time (without the wait time) ... we will revisit this later.
  - From O.S. class, we use: user time, real time, sys time.

$$Elapse\ Time = CPU\ Time + I/O\ Time + Memory\ Time$$

- ★ Throughput
  - Tasks per time unit
- ★ Units for commercialization (Not a real performance)
  - MIPS
    - Millions of instructions per second
  - FLOPS
    - Floating Point Operations per second

- FLOPs is not a good measure of performance



https://arxiv.org/abs/2101.09336

EfficientNetv2

- Architecture/hardware aware NAS + handcrafted building blocks (from mobilenetV2)
- Tradeoff between height, width, depth, resolution

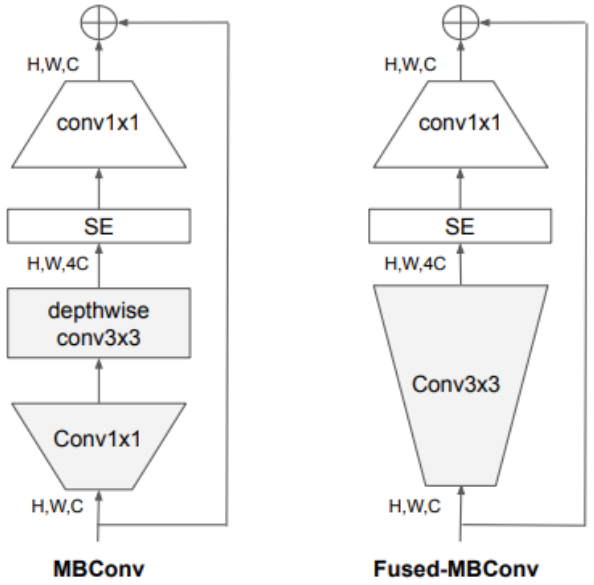


Figure 2. Structure of MBConv and Fused-MBConv.

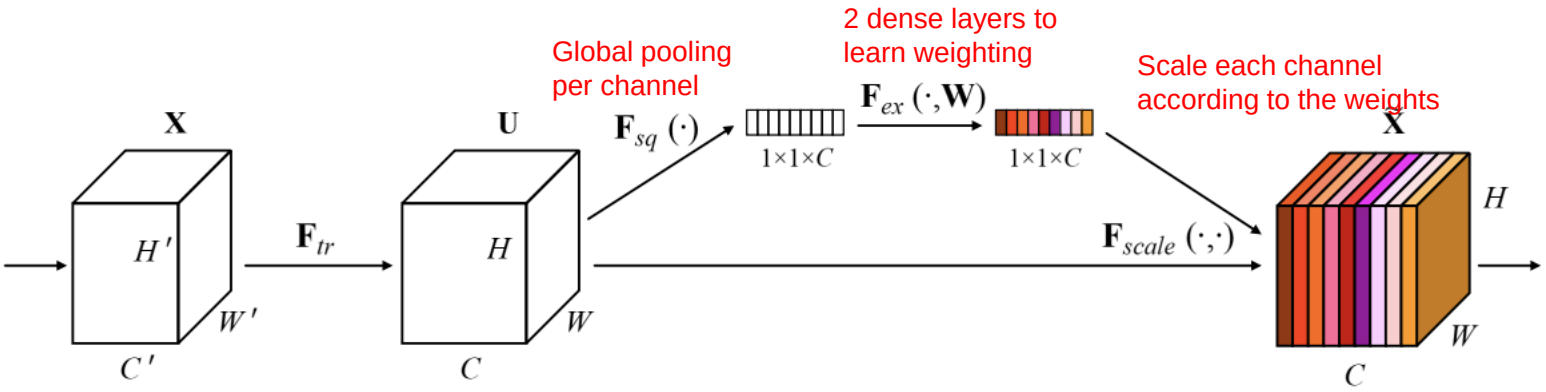
https://arxiv.org/abs/1905.11946
https://arxiv.org/abs/2104.00298

Modern convolutional blocks

- Squeeze and excitation
- Mobile inverted bottleneck

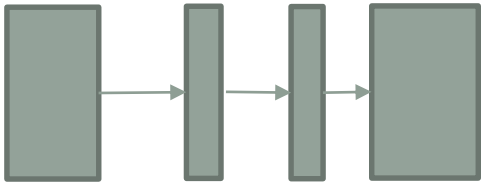
Squeeze and excitation

- Maybe each channel should be weighted differently?
  - A picture with wave patterns should look for fish features
  - A picture with grass patterns should look for cow features
- “Attention” on channel info (no softmax nor weighted sum)

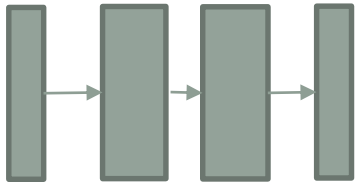


Inverted bottleneck

- A bottleneck block
  - reduce channels using 1x1
  - Regular conv on fewer channels
  - Increase channel using 1x1



- Inverted bottleneck
  - Increase channels using 1x1
  - Regular conv on more channels
  - Reduce channels using 1x1



- Bottleneck combines information between channels
- With residual connections, inverted bottleneck uses less runtime memory during inference (DAG computation graph)

https://arxiv.org/abs/1801.04381

Mobile inverted Bottleneck

- Use depthwise convo instead of conv (MobileNetv2)
- Add SE (EfficientNetv1)
- Can fused 1x1 with depthwise (EfficientNetv2)

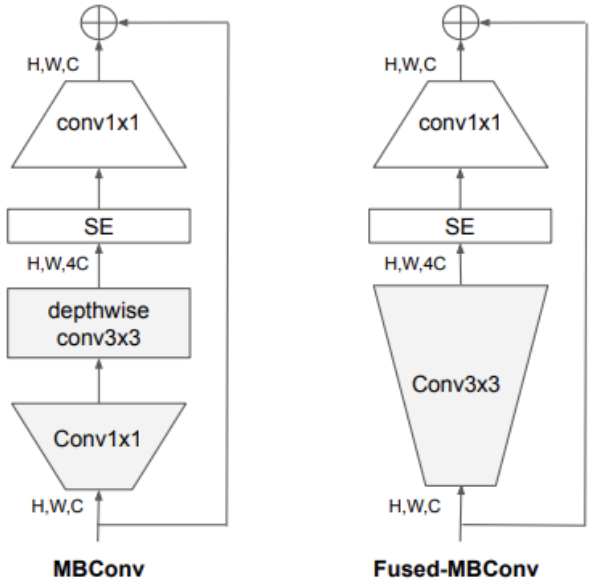
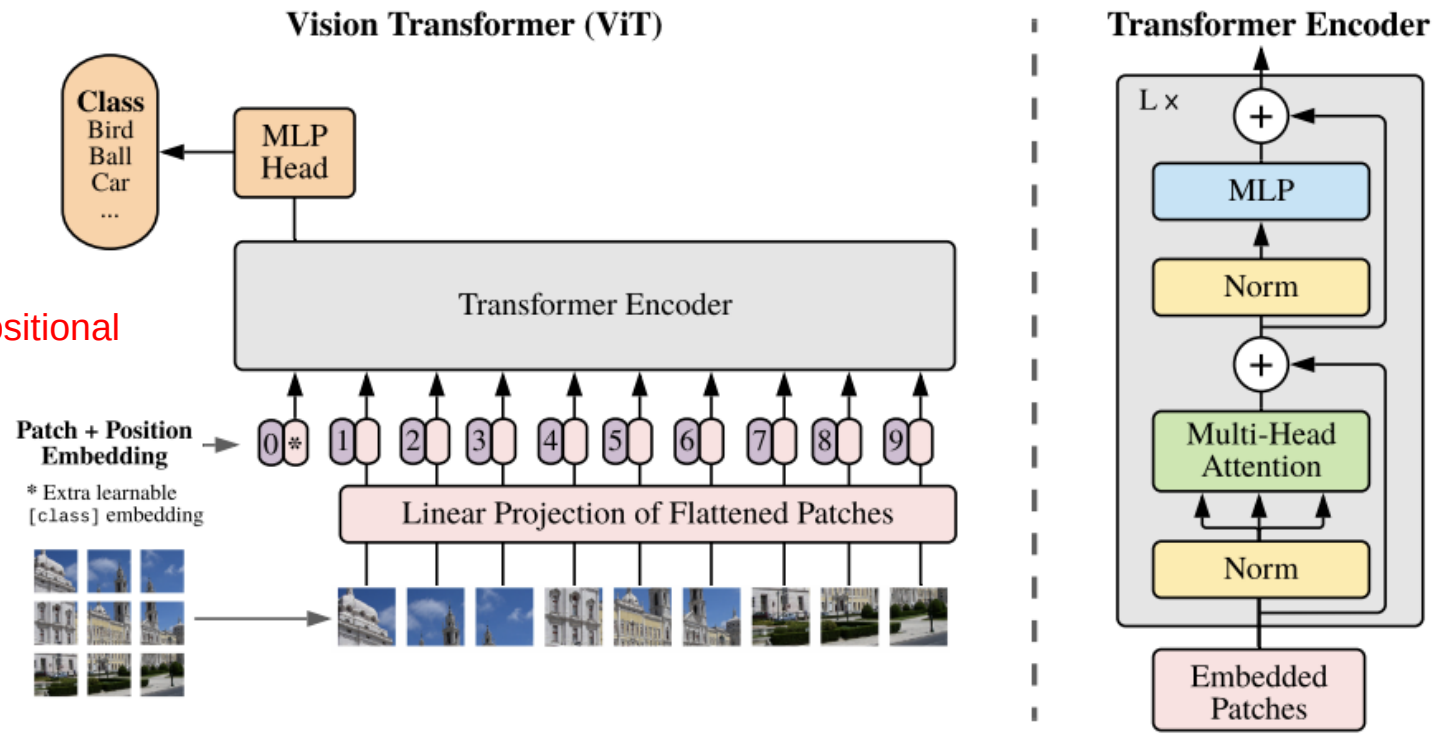


Figure 2. Structure of MBConv and Fused-MBConv.

Vision transformer

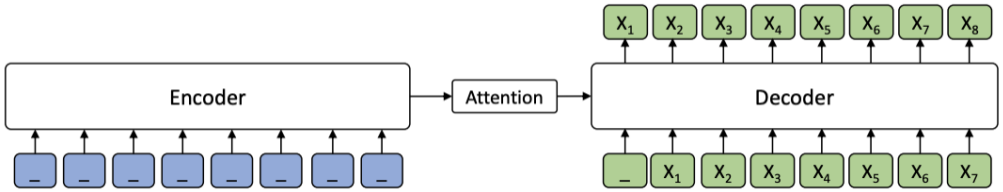
The paper explains this model using less than one page



Learnable positional embedding

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
https://arxiv.org/pdf/2010.11929v1.pdf

Dall E



GPT3 for images
Cluster image patches into tokens and combined with text (BPE) into a single token to train an autoregressive model (GPT-like)

(a) a tapir made of accordion. a tapir with the texture of an accordion.

(b) an illustration of a baby hedgehog in a christmas sweater walking a dog

(c) a neon sign that reads "backprop". a neon sign that reads "backprop". backprop neon sign

(d) the exact same cat on the top as a sketch on the bottom

Figure 2. With varying degrees of reliability, our model appears to be able to combine distinct concepts in plausible ways, create anthropomorphized versions of animals, render text, and perform some types of image-to-image translation.

https://openai.com/blog/dall-e/
https://arxiv.org/pdf/2102.12092.pdf

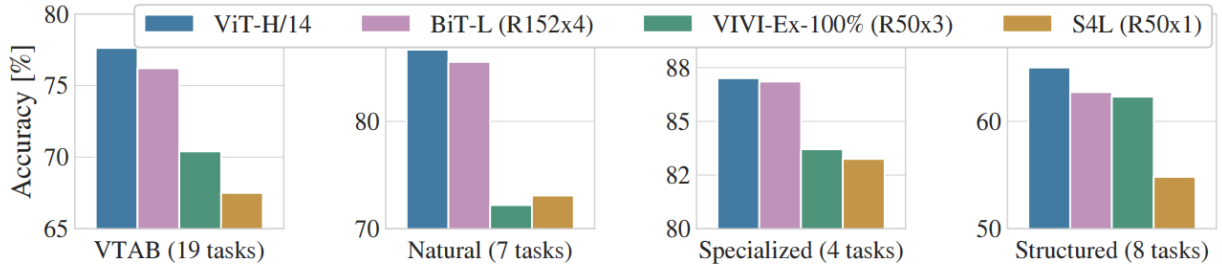


Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

Only wins if data is VERY large

Does not work well for object detection and semantic segmentation as backbones

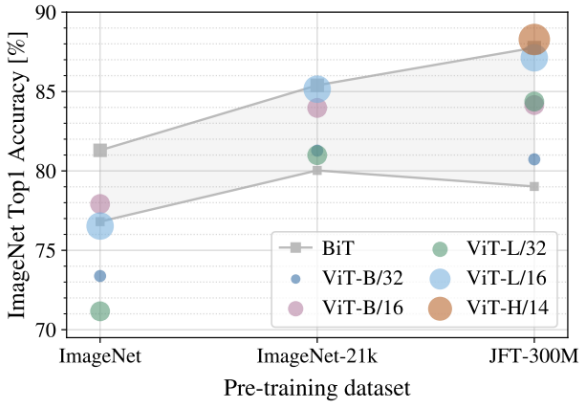


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

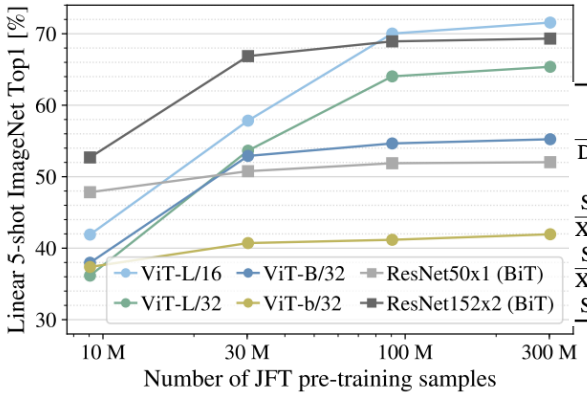


Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

https://arxiv.org/pdf/2103.14030.pdf

Swin Transformer

- So…we're dealing with images
  - Different scale
  - Translation equivalence (object detection)
    - Translation invariance – move input, same output
    - Translation equivariance – move input, move output same amount

Swin Transformer

- Multi-scale
- Local attention (conv-like) in a local window
- Weights across local windows are shared
  - Translation equivariance

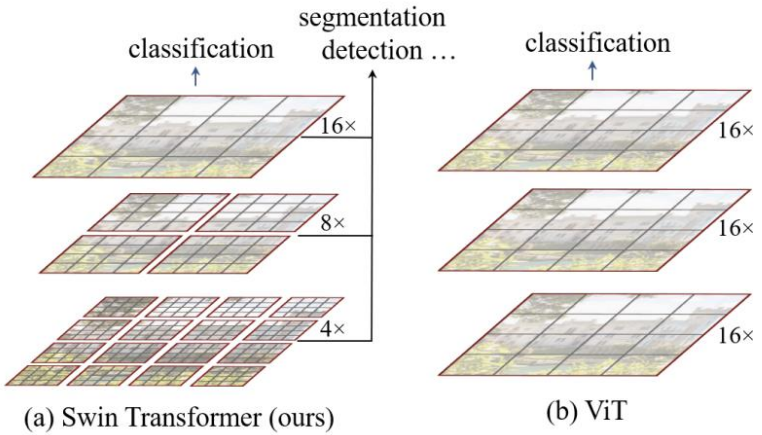
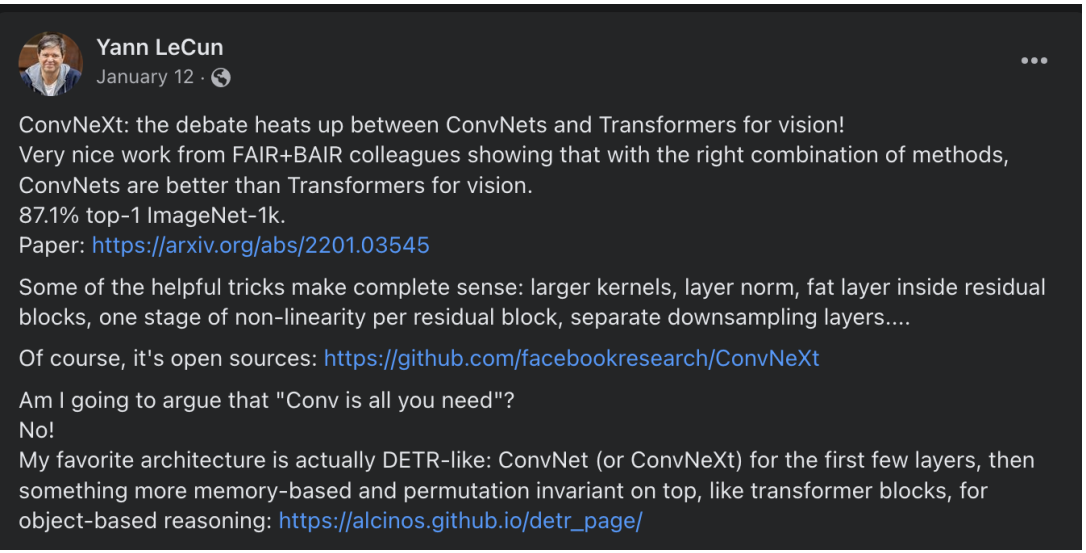


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

ConvNext



Yann LeCun
January 12 ·

ConvNeXt: the debate heats up between ConvNets and Transformers for vision!
Very nice work from FAIR+BAIR colleagues showing that with the right combination of methods, ConvNets are better than Transformers for vision.
87.1% top-1 ImageNet-1k.
Paper: https://arxiv.org/abs/2201.03545

Some of the helpful tricks make complete sense: larger kernels, layer norm, fat layer inside residual blocks, one stage of non-linearity per residual block, separate downsampling layers....

Of course, it's open sources: https://github.com/facebookresearch/ConvNeXt

Am I going to argue that "Conv is all you need"?
No!
My favorite architecture is actually DETR-like: ConvNet (or ConvNeXt) for the first few layers, then something more memory-based and permutation invariant on top, like transformer blocks, for object-based reasoning: https://alcinos.github.io/detr_page/

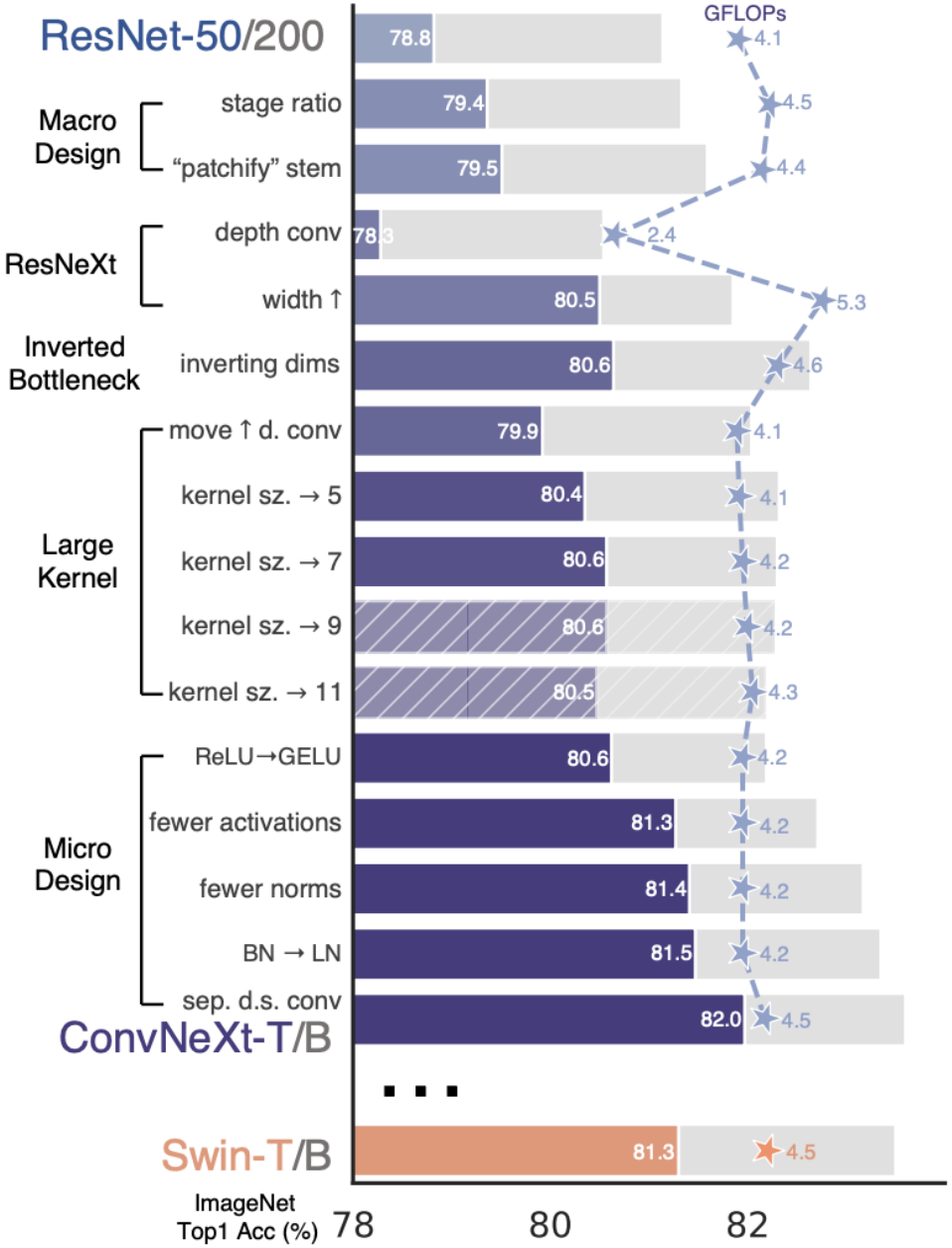
The full dominance of ConvNets in computer vision was not a coincidence: in many application scenarios, a "sliding window" strategy is intrinsic to visual processing, particularly when working with high-resolution images. ConvNets have several built-in inductive biases that make them well-suited to a wide variety of computer vision applications. The most important one is translation equivariance, which is a desirable property for tasks like objection detection. ConvNets are also inherently efficient due to the fact that when used in a sliding-window manner, the computations are shared [62].

Translation invariance – move input, same output
Translation equivariance – move input, move output same amount

https://arxiv.org/pdf/2201.03545.pdf

ConvNext

- To get the most recent hype see ConvNext
- Bag of tricks++
- Highly recommended as a review paper



https://arxiv.org/abs/2201.03545

Object classifiers summary

Most successful tricks:

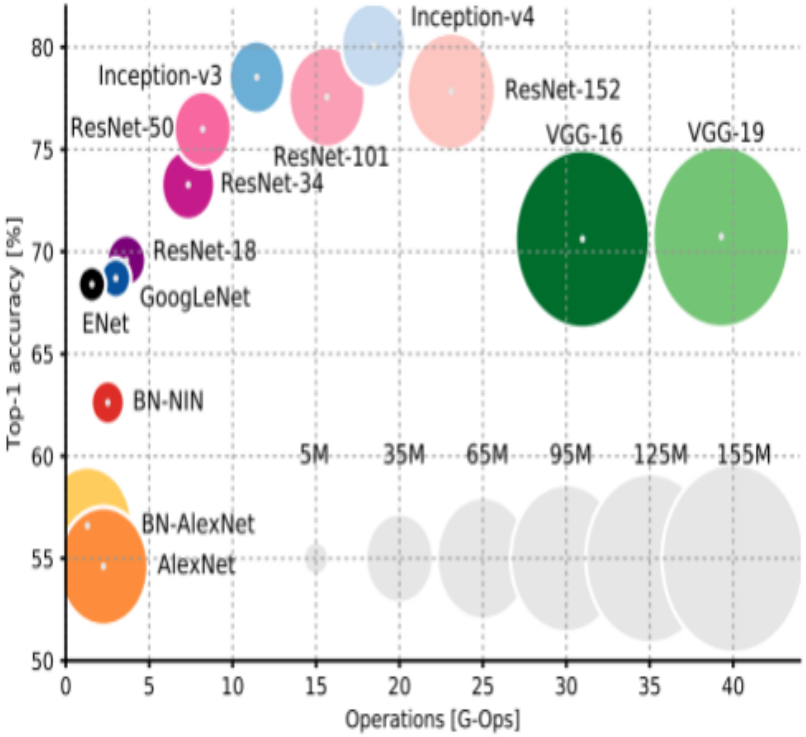
residual, batch norms (layer norms), multi-path, data augmentation

These object classifiers are often called backbones and used by other models

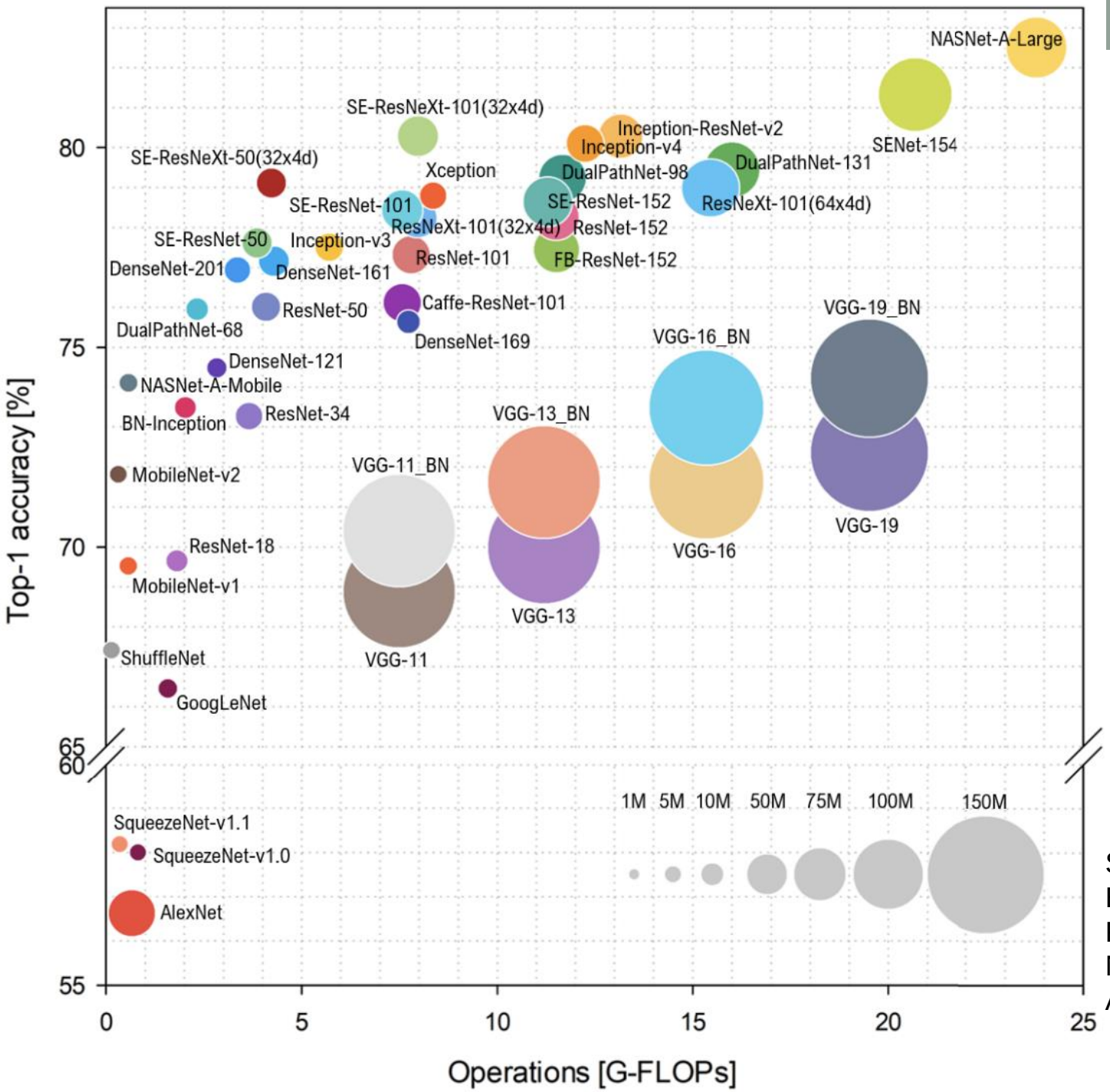
Feature Pyramid is a neck

Output layer is the head

Comparing accuracy, model size, transferability and compute.



Alfredo Canziani, et al. “An analysis of deep neural network models for practical applications,” 2017



Simone Bianco, et al., Benchmark Analysis of Representative Deep Neural Network Architectures, 2019

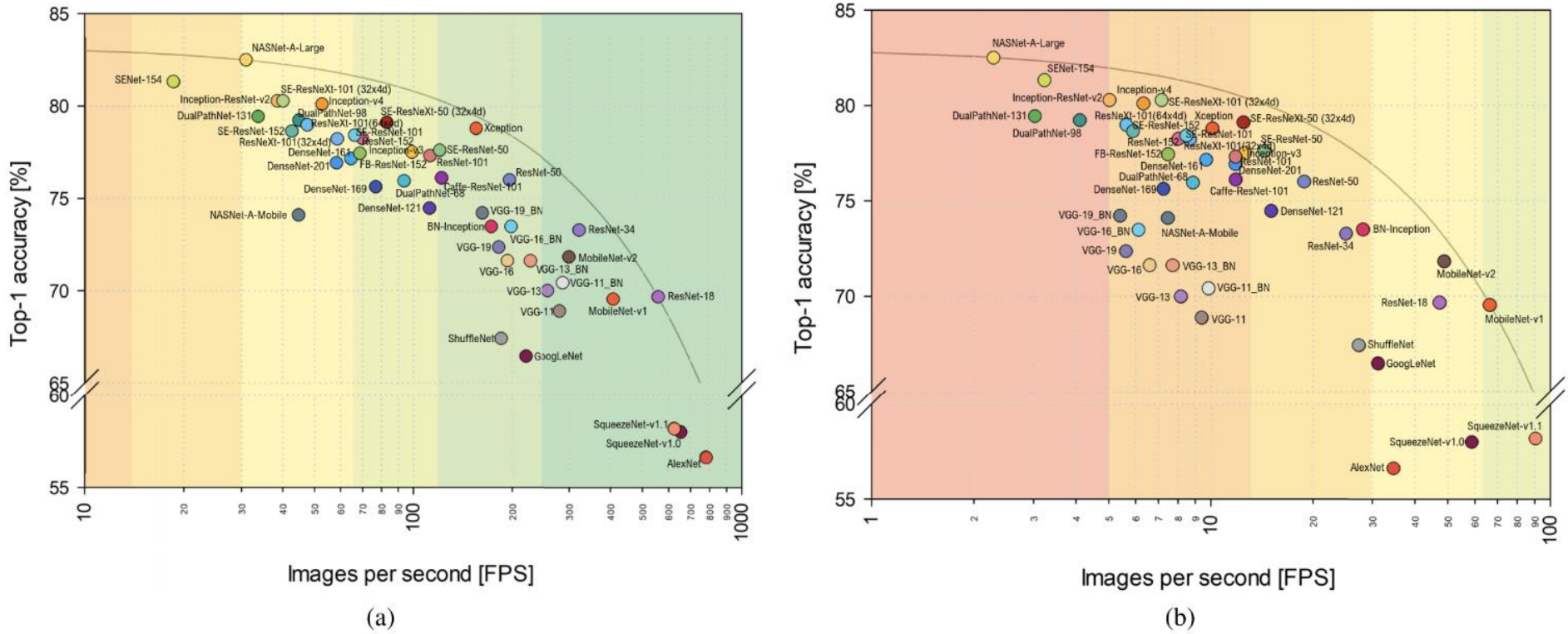


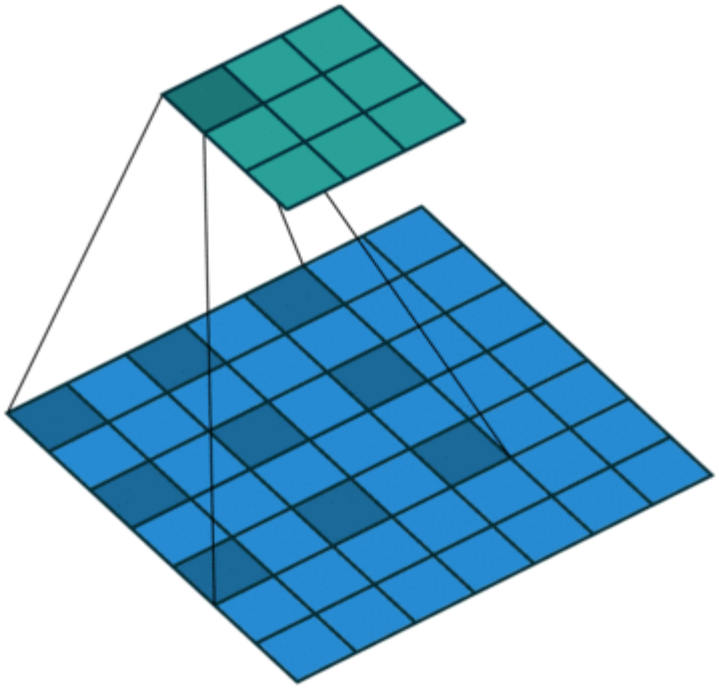
FIGURE 3. **Top-1 accuracy *vs.* number of images processed per second (with batch size 1) using the Titan Xp (a) and Jetson TX1 (b).**

Simone Bianco, et al., Benchmark Analysis of Representative Deep Neural Network Architectures, 2019

Other stuff to look for

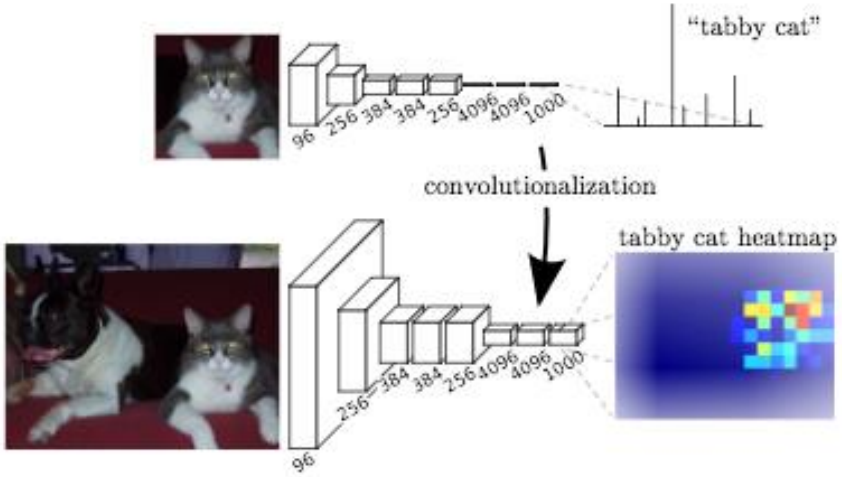
Layers

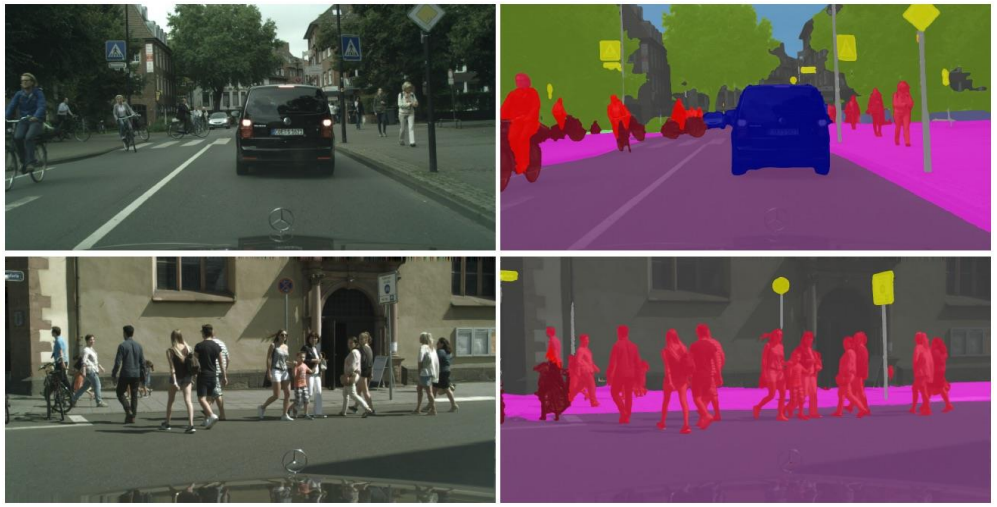
Dilated convolution https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25



Sometimes you want to increase the size of your feature map

Image segmentation



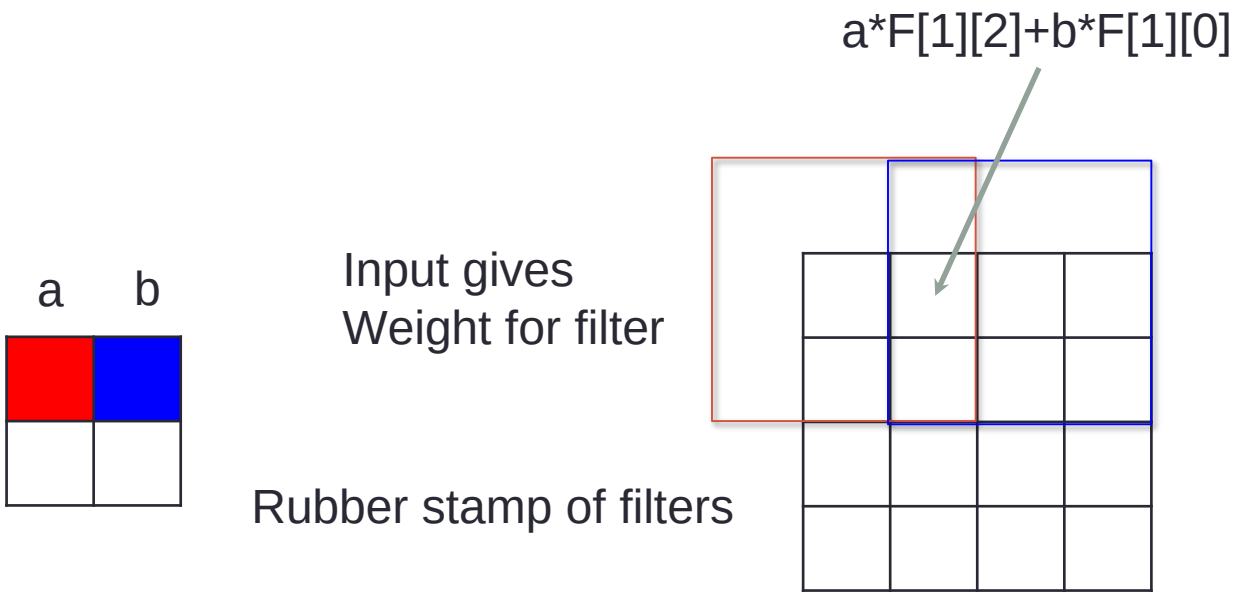


2 main approaches to upsample
De-convolution
resize (unpooling) +convolution

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
http://vladlen.info/publications/feature-space-optimization-for-semantic-video-segmentation/

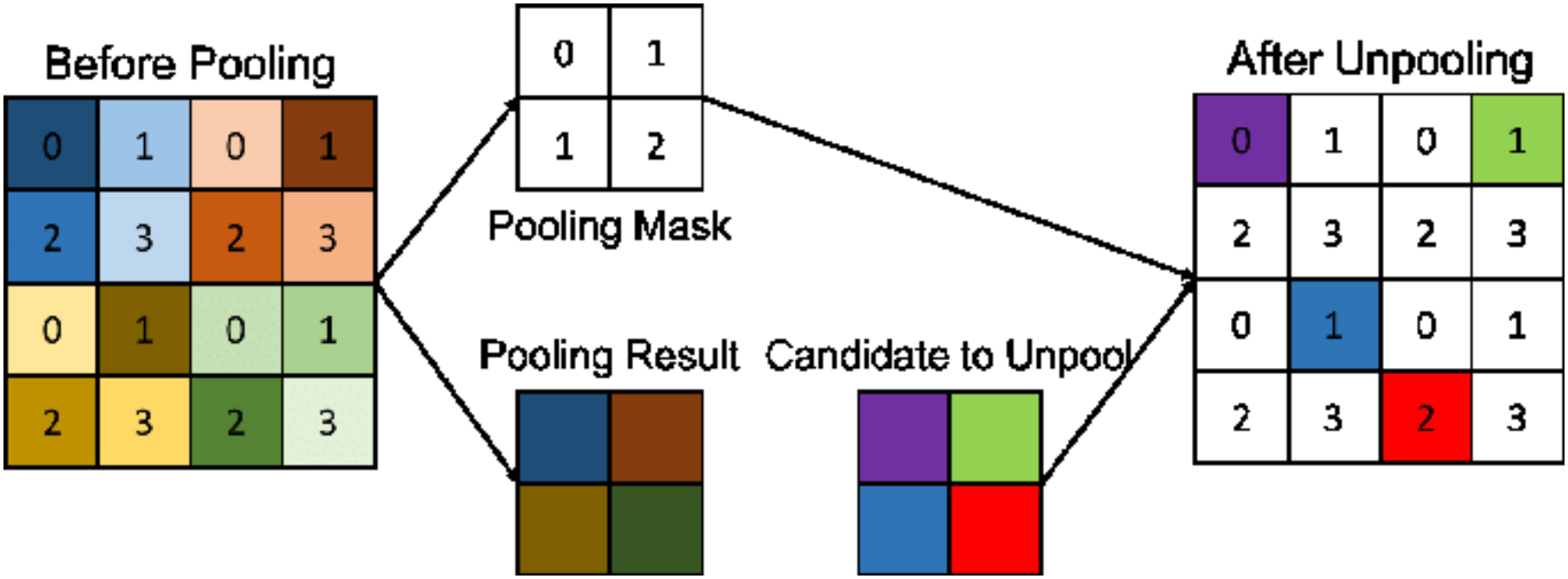
De-convolution (Upsampling)

- 3x3 de-convolution filter, stride 2, pad 1



Other names because this name sucks (for me)
- Convolution transpose, upconvolution, backward strided convolution

Unpooling + conv



Remember the location of the max value

Put the value to unpool at that location

Fill the rest with zeroes

Follow by regular convolution to smooth the image

Fully Automatic Acute Ischemic Lesion Segmentation in DWI Using Convolutional Neural Networks, 2017.

Upsampling notes

Deconvolution filter size should be a multiple of the stride to avoid checkerboard artifacts

Unpooling can be replaced with regular image resizing techniques (interpolation/upsample

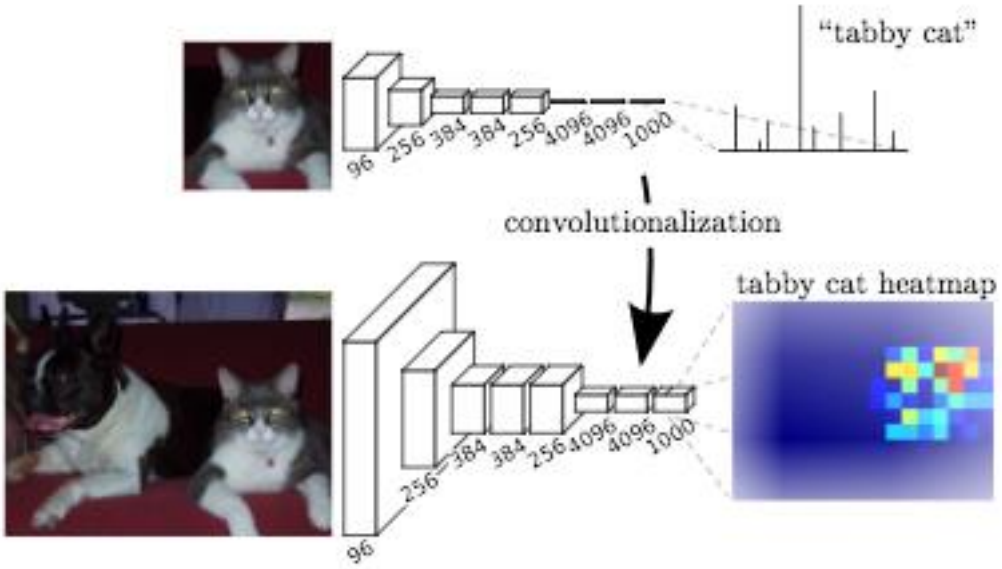
https://pytorch.org/docs/stable/generated/torch.nn.Upsample.html)

Image using deconv

Image using resize upsampling

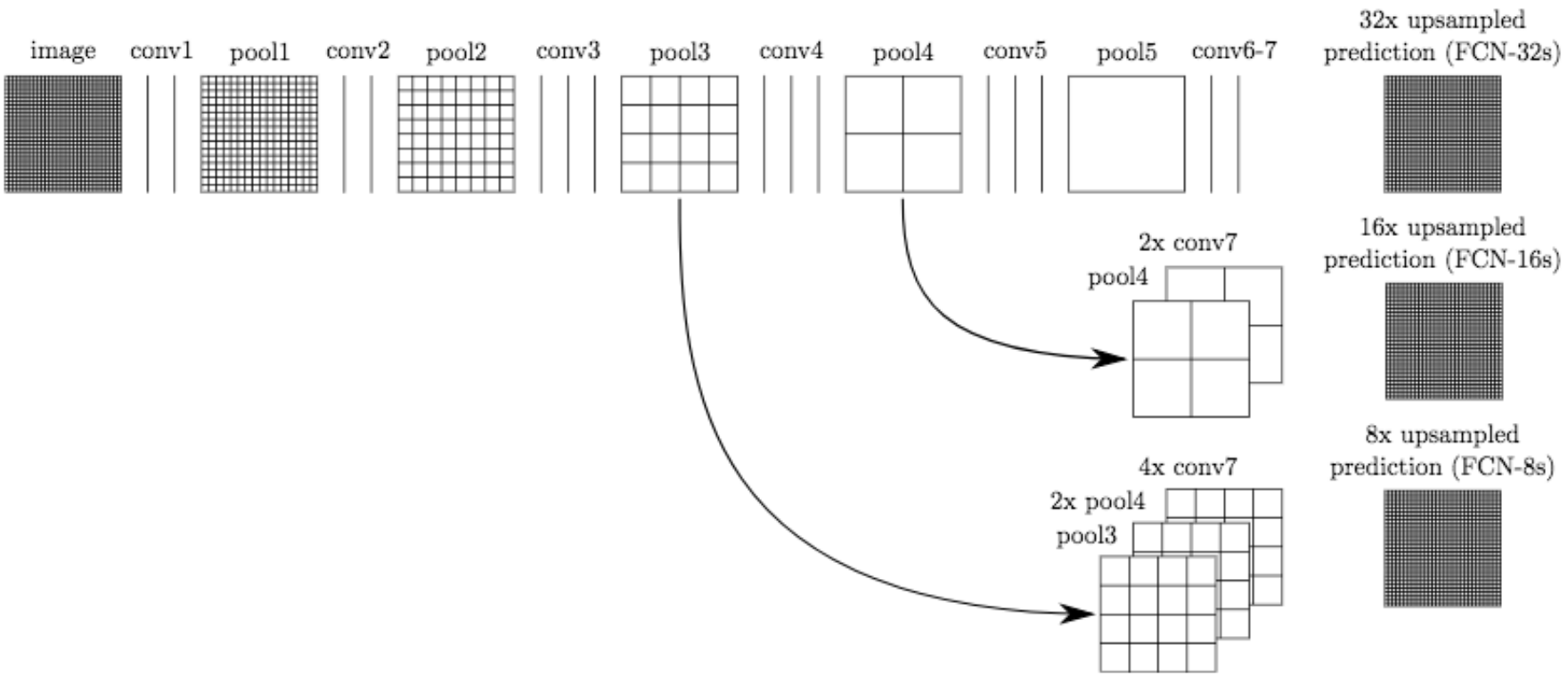
https://distill.pub/2016/deconv-checkerboard/

De-convolution for segmentation



https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

De-convolution for segmentation



https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

De-convolution for segmentation

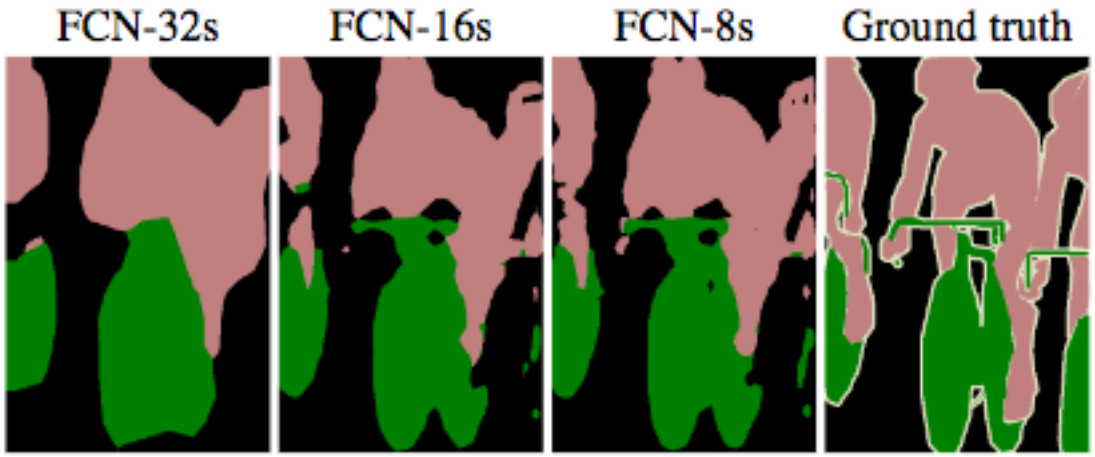


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Unet

- A typical architecture for segmentation task
- Have a skip connection to preserve resolution

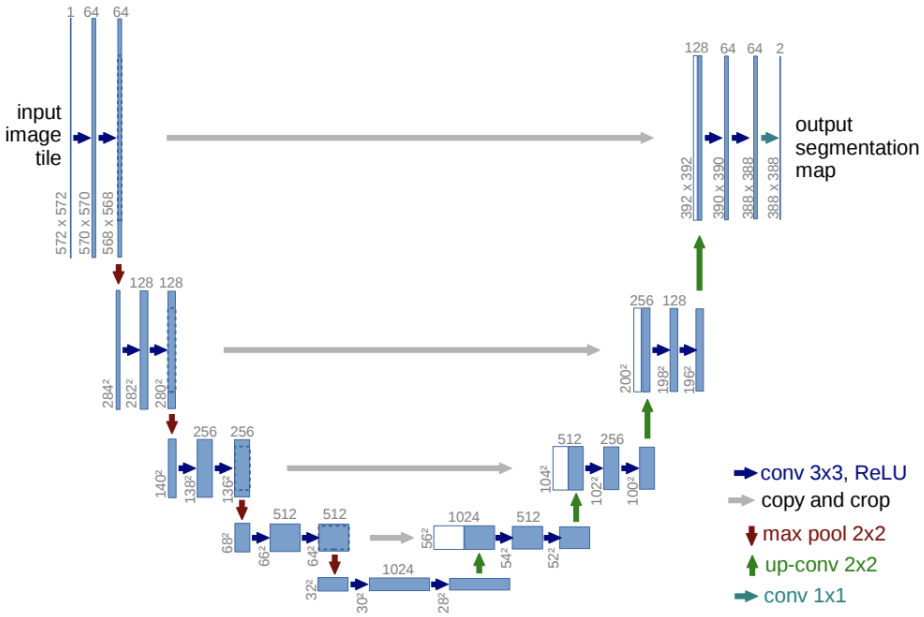


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Guide on choosing the architecture

- Is it solved before? -> Copy!
- Is it similar to some standard vision/NLP task? -> copy with modification
- Nothing close?
  - What would a human do?
  - What kind of inductive bias should we put in?
    - What kind of invariance does the data has?
    - What kind of noise/augmentation would work?
- Start simple then add extra
- Overfit then think of ways to regularize

Implementation

- Codes usually written as blocks

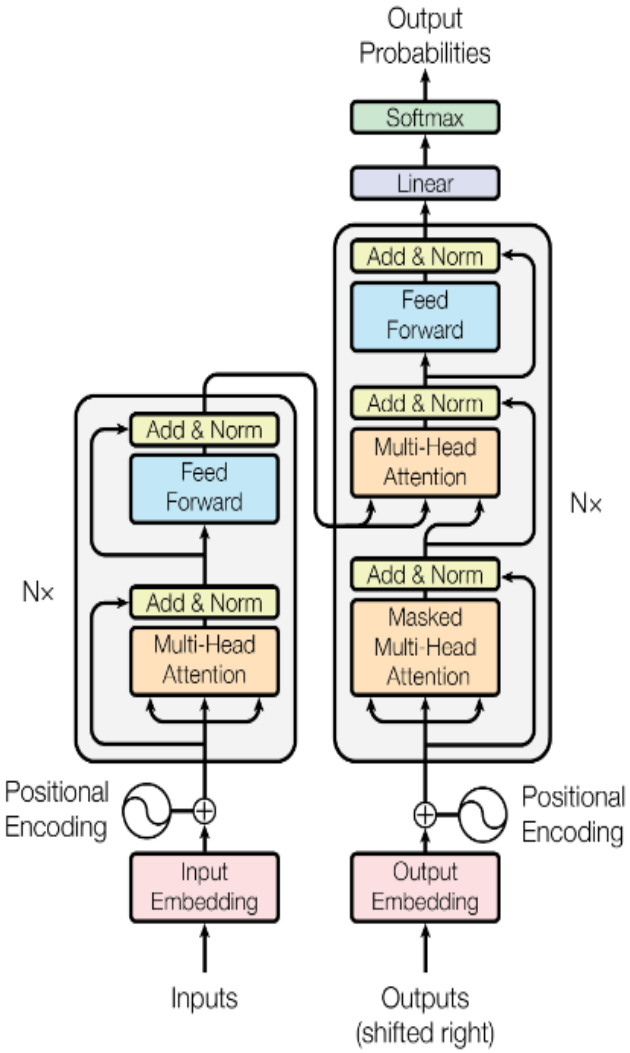
https://github.com/milesial/Pytorch-UNet/

```python
class UNet(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=False):
        super(UNet, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = DoubleConv(n_channels, 64)
        self.down1 = Down(64, 128)
        self.down2 = Down(128, 256)
        self.down3 = Down(256, 512)
        factor = 2 if bilinear else 1
        self.down4 = Down(512, 1024 // factor)
        self.up1 = Up(1024, 512 // factor, bilinear)
        self.up2 = Up(512, 256 // factor, bilinear)
        self.up3 = Up(256, 128 // factor, bilinear)
        self.up4 = Up(128, 64, bilinear)
        self.outc = OutConv(64, n_classes)

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
```

```python
class Up(nn.Module):
    """Upscaling then double conv"""

    def __init__(self, in_channels, out_channels, bilinear=True):
        super().__init__()

        # if bilinear, use the normal convolutions to reduce the number of channels
        if bilinear:
            self.up = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)
            self.conv = DoubleConv(in_channels, out_channels, in_channels // 2)
        else:
            self.up = nn.ConvTranspose2d(in_channels, in_channels // 2, kernel_size=2, stride=2)
            self.conv = DoubleConv(in_channels, out_channels)

    def forward(self, x1, x2):
        x1 = self.up(x1)
        # input is CHW
        diffY = x2.size()[2] - x1.size()[2]
        diffX = x2.size()[3] - x1.size()[3]

        x1 = F.pad(x1, [diffX // 2, diffX - diffX // 2,
                        diffY // 2, diffY - diffY // 2])
        # if you have padding issues, see
        # https://github.com/HaiyongJiang/U-Net-Pytorch-Unstructured-Buggy/commit/0e854509c2cea854
        # https://github.com/xiaopeng-liao/Pytorch-UNet/commit/8ebac70e633bac59fc22bb5195e513d5832
        x = torch.cat([x2, x1], dim=1)
        return self.conv(x)
```

The architectures lectures

- Basic pieces
  - Dense
  - Conv
  - RNNs
  - Attention
  - Residual connection
  - Upsampling, Deconv
- Blocks
  - Squeeze and Excite
  - Transformer
- Tricks
  - BN, LN
  - Augmentation



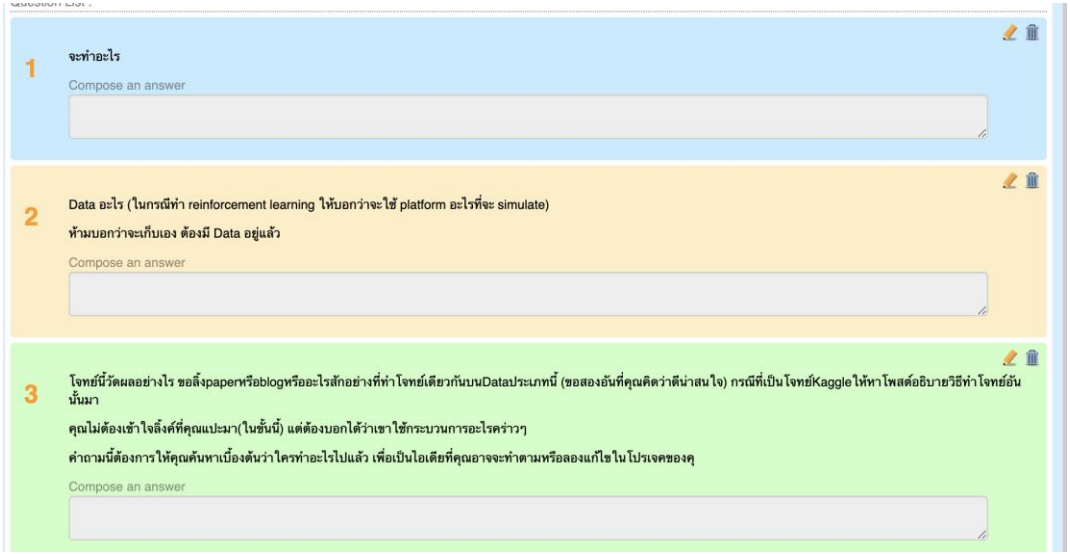
Later lectures will focus on other things beside the architecture

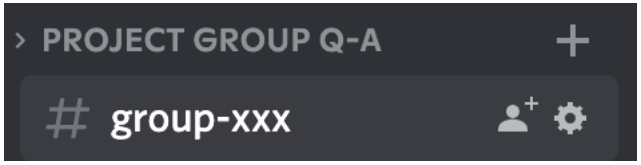
Course project

- <= 5 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

Proposal method

- Answer the questions





- Then, we'll have back and forth comments in discord

GTC is happening!

- **Register free for NVIDIA GTC by clicking here.**
- **^ Use the above refer link please**
- **Watch one session and get 1% extra credit**

Deep learning sessions

- · S42629 AI Innovations: Three Case Studies on Successful Natural Language Processing (Presented by Oracle Cloud)
- · S41564 Conversational AI for the Next 500 million?
- · S41735 P-tuning: An Effective Prompt Engineering Method to Significantly Improve the Performance of Your Large NLP Model
- · S41421 Faster Neural Network Training, Algorithmically
- · S42557 Representation Learning and Evaluation for Interaction Tasks

Healthcare

- · S41186 Cancer Diagnostics Enabled by Machines: Finding and Classifying Single Tumor Cells in Human Tissue
- · S42061 Medical Image Reconstruction with Memory-efficient Neural Networks
- · SE1991 NVIDIA FLARE: An Open Federated Learning Platform
- · S42036 Accelerating Healthcare Innovation with AI
- · S41710 Capitalizing on the Metadata of Medical Imaging to boost AI Performance:

Omniverse

- · S42582: Collaborative Design and the Future of Advertising: WPP and Lenovo -Omniverse fireside chat
- · S41636: Modernizing Product Design, Simulation, and Direct-to-consumer Manufacturing Workflows, all within Omniverse
- · SE2310 Meet the Omniverse Experts: Materials and Rendering Technology in Omniverse
- · S42109 Creating a Real-time, Photoreal Ramen Shop in Omniverse
- · S41591 How to Build Extensions and Apps on Omniverse Kit

- Deep Dive: One Click Animation Retargeting in Omniverse [S41482]
- How to Design Collaborative AR and VR worlds in Omniverse [S42162]
- Populating Virtual Worlds for Simulation and AI Training [S41746]
- Developer Breakout: Building Extensions and Apps on Omniverse [SE2305]
- Developer Breakout: Build Your Own Microservices on Omniverse [SE2306]
- Meet the Omniverse Experts: Materials and Rendering Technology in Omniverse [SE2310]
- How to Build Extensions and Apps on Omniverse Kit [S41591]
- Leveraging NVIDIA Omniverse for simulation and collaboration for a variety of wireless communication use cases [SE2683]
- Creating a Real-time, Photoreal Ramen Shop in Omniverse [S42109]
- Bring It to Life: A Look at Omniverse's New Runtime Animation System [S42135]
- A Primer on Materials for NVIDIA Omniverse [S41192]
- Breakout Session: Building Industrial Digital Twins on Omniverse [SE2311]
- Toward Bi-directional Workflows with Omniverse and ArcGIS CityEngine [S41731]
- Breakout Session: Exploring Omniverse for Manufacturing [SE2304]
- Maximizing Hardware Resources and Flexibility in Omniverse using vGPU in the Data Center [S41727]
- Connect with the Experts: AR and VR in Omniverse [CWE41667]
- Industry Breakout: Omniverse for Media & Entertainment [SE2318]
- Architectural Design, Omniverse, and the Simulated World [S41653]

- Industry Breakout: Omniverse for Media & Entertainment [SE2318]
- Architectural Design, Omniverse, and the Simulated World [S41653]
- Making a Connector for Omniverse [S41592]
- Simulating and Optimizing a Warehouse Operation Based on Omniverse [S41507]
- Understanding Wildfire Behavior using Machine Learning and Omniverse (Presented by Lockheed Martin) [S41739]
- Toward a Synchronized Manufacturing Digital Twin: NVIDIA Omniverse has the Building Blocks [S41632]
- Synthetic Data Generation using the Omniverse Replicator SDK [S41581]
- Towards real-time fusion reactor design using Omniverse [S41699]
- Omniverse Opportunities for Startups [S41872]
- AI-powered Infrastructure Digital Twin with Omniverse [S41836]
- New Era of Digital Twins with Omniverse [SE2644]
- Transforming Global Film Production Workflows with Omniverse [S42688]
- Omniverse and Virtual Workstations: The End-user Experience Results Are In [S41146]
- Artist Breakout: NVIDIA Studio and Omniverse for the Next Era of Creativity [SE2307]
- Simulating Accessibility with Omniverse: The Future of Inclusive Design [S42117]
- Building Digital Twins of the Earth in Omniverse [S41939]
- Emerging Venture Themes for 2022: Omniverse and Metaverse [S42322]
- NVIDIA Omniverse for Architecture, Engineering, Construction, and Operations [SE2271]
- Isaac Cortex: A Decision Framework for Virtual and Physical Robots [S42693]