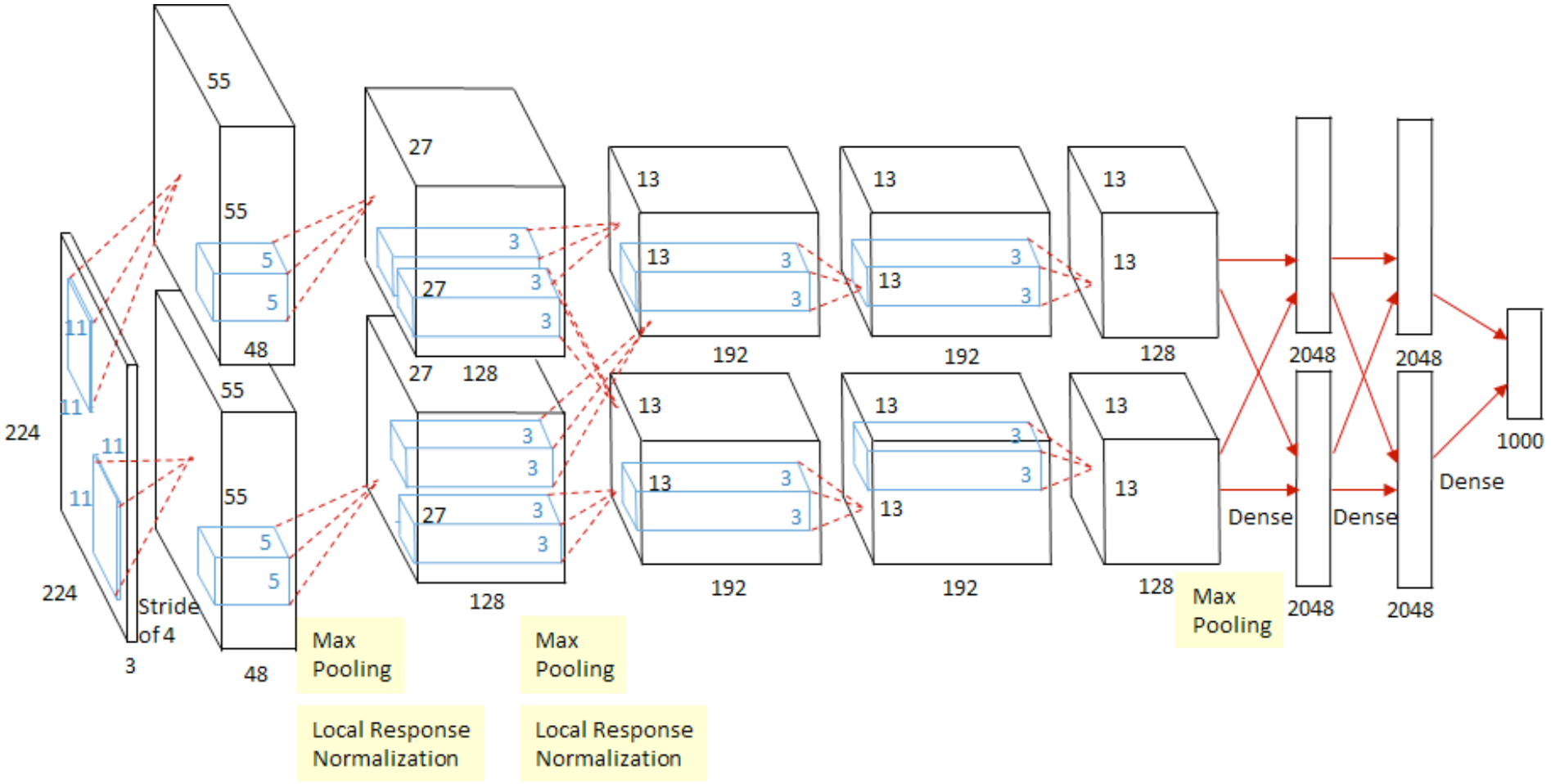
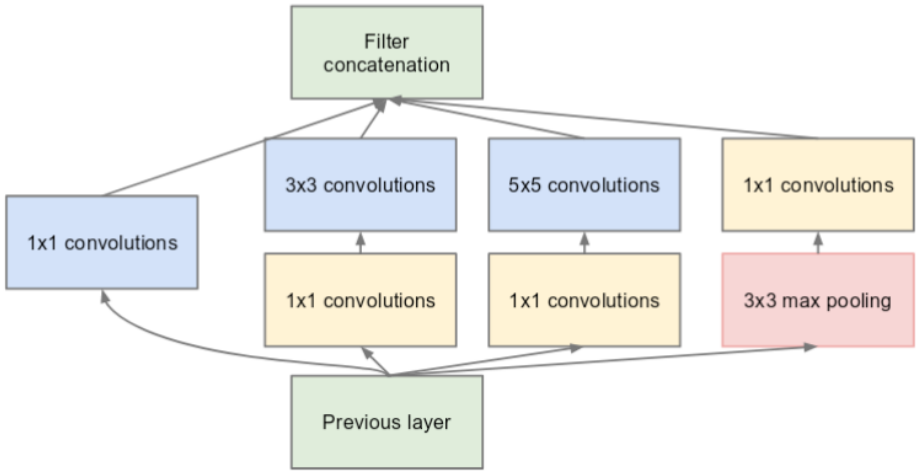
Whirlwind tour of modern deep learning

Vision Model

Alexnet (2012) - Conv. +GPU

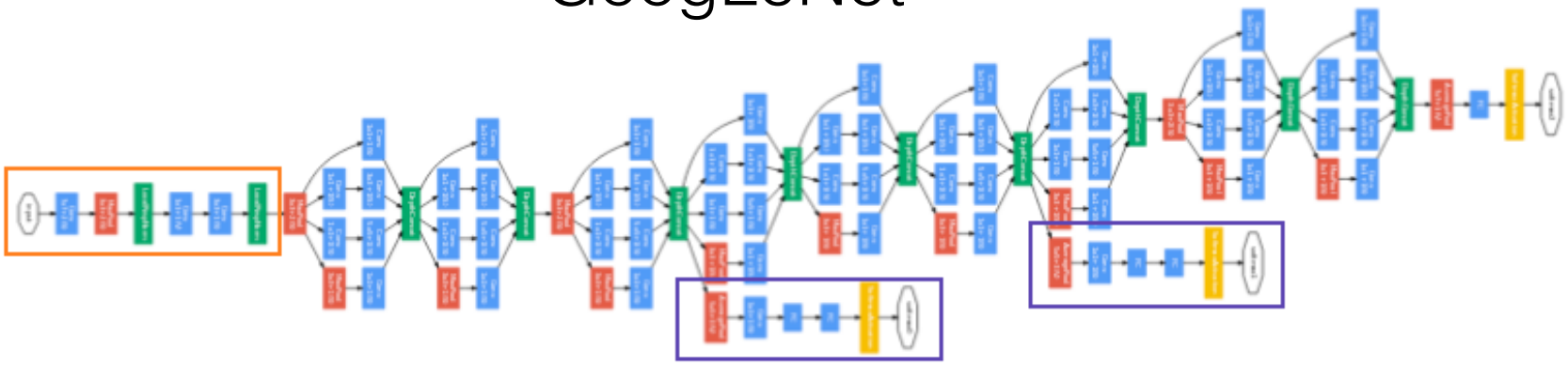


Inception 2014-2016

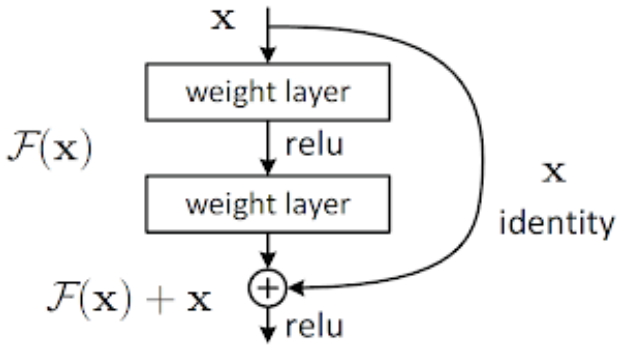


(b) Inception module with dimension reductions

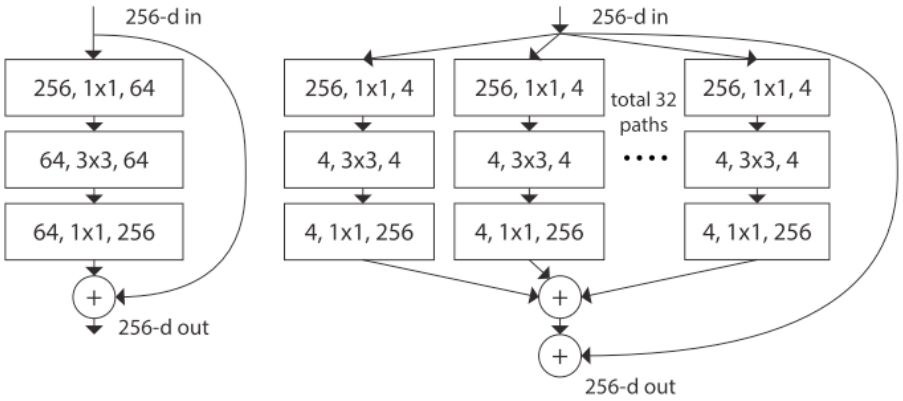
GoogLeNet

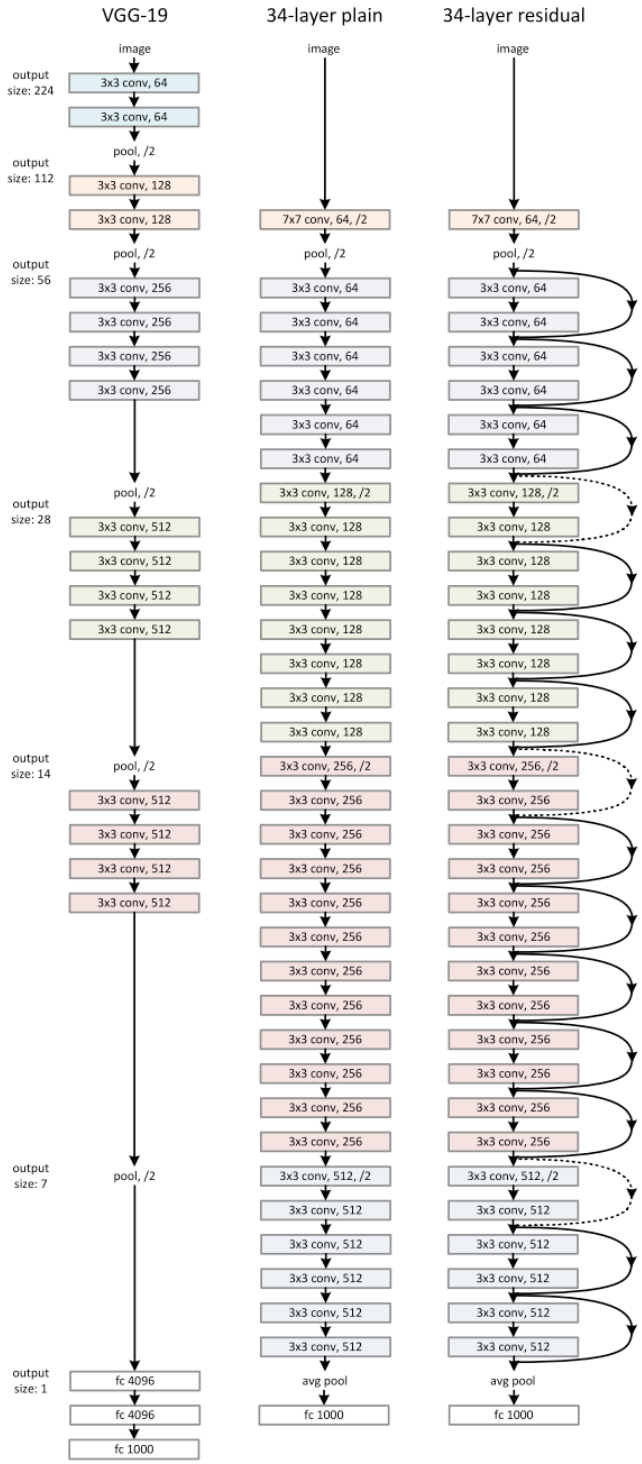


Resnet 2016



ResXnet 2016





TORCHVISION.MODELS

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet = models.mobilenet_v2(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are expected to be at least 224. The images have to be loaded in to a range of [0, 1] and then normalized using `mean = [0.485, 0.456, 0.406]` and `std = [0.229, 0.224, 0.225]`. You can use the following transform to normalize:

```
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
```

https://keras.io/api/applications/#usage-examples-for-image-classification-models

Neural style transfer

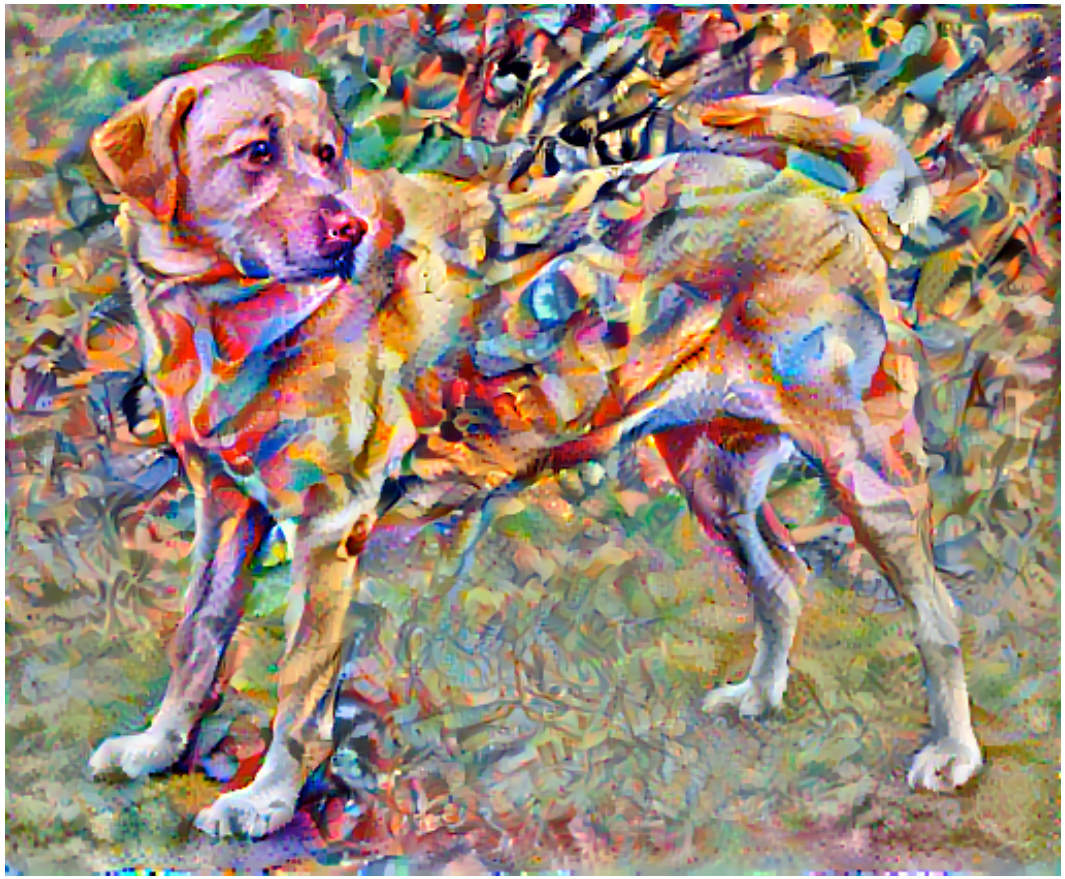
https://www.tensorflow.org/tutorials/generative/style_transfer

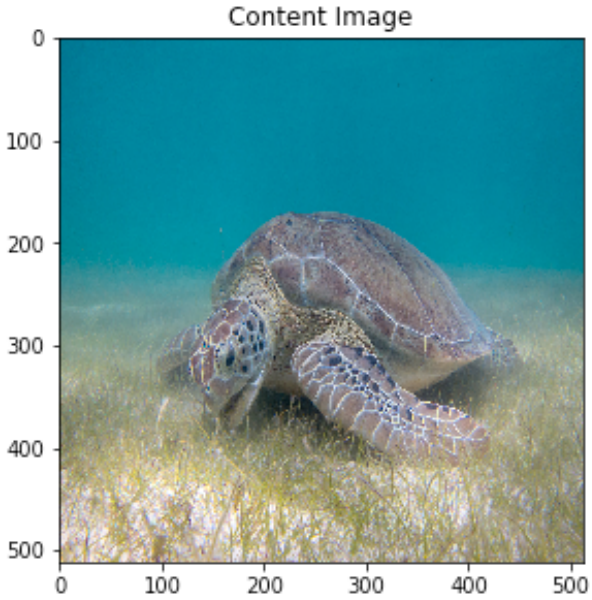
https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398

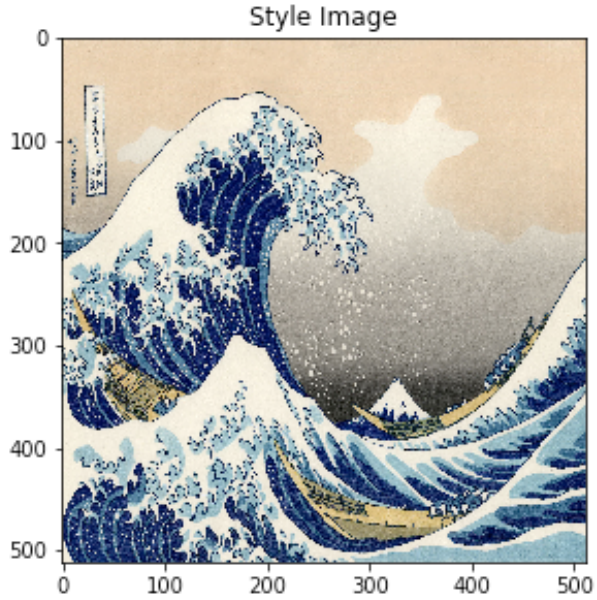
Content

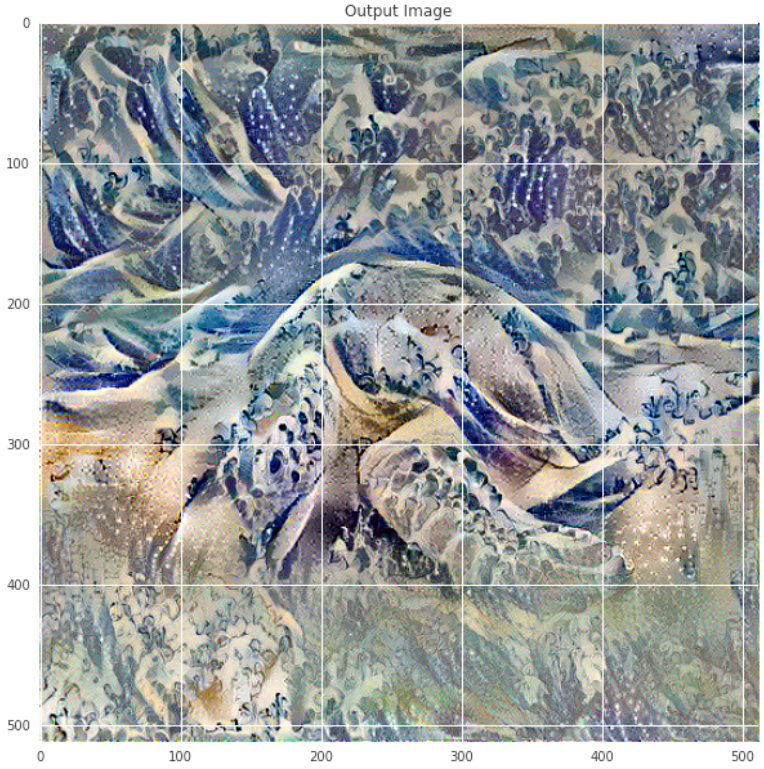
Style

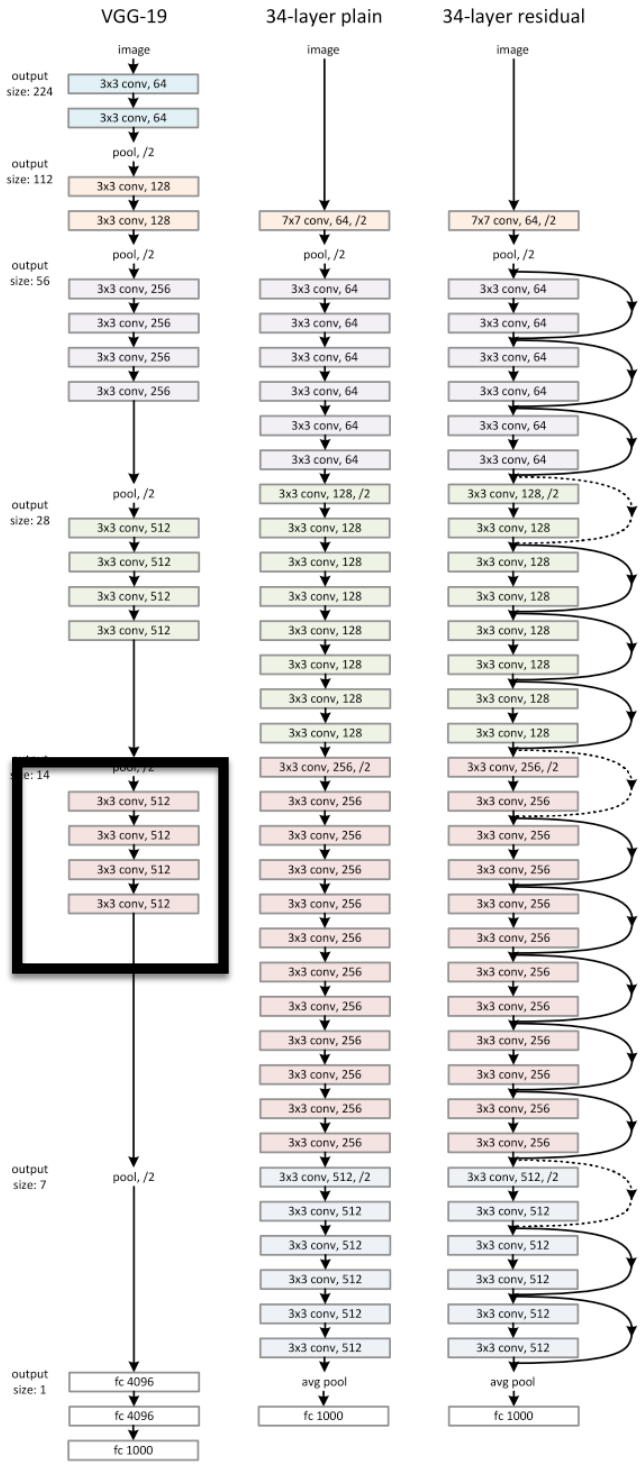
+




Content Image
0
100
200
300
400
500
0
100
200
300
400
500


Style Image
0
100
200
300
400
500
0
100
200
300
400
500


Output Image
0
100
200
300
400
500
0
100
200
300
400
500



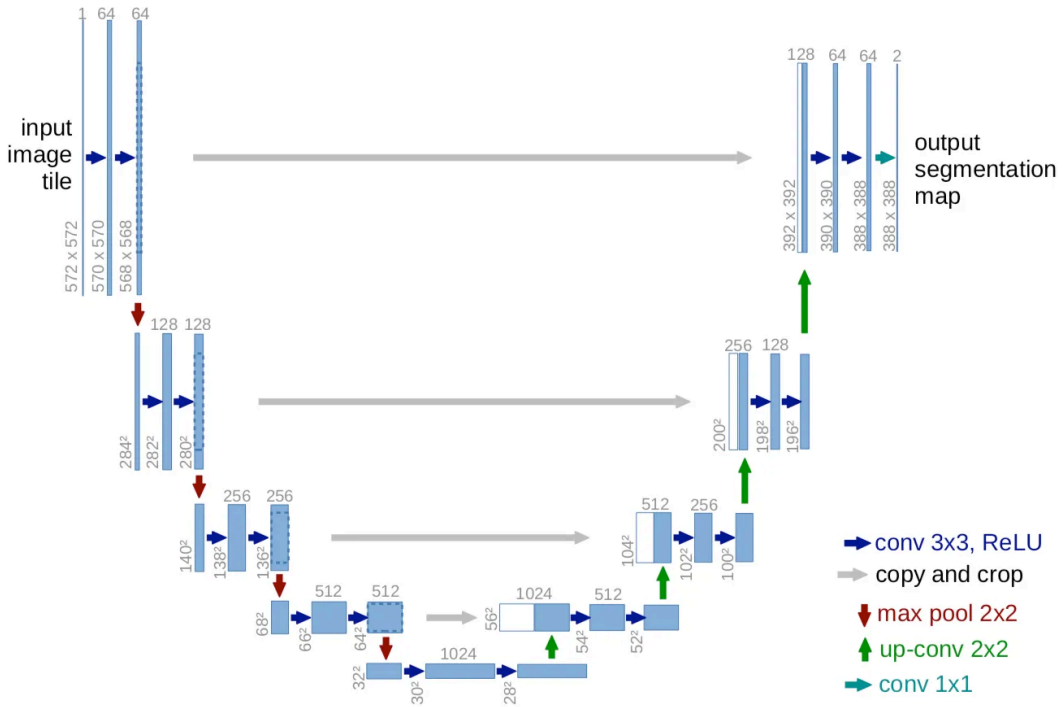
Content: means

Style: Correlations between filter

$$L^l_{content}(p, x) = \sum_{i,j}(F^l_{ij}(x) - P^l_{ij}(p))^2$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j}(G^l_{ij} - A^l_{ij})^2$$

$$L_{style}(a, x) = \sum_{l \in L} w_l E_l$$



https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5

https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

Natural Language Processing

Word2Vec: Map words to vector space

https://www.tensorflow.org/tutorials/text/word2vec

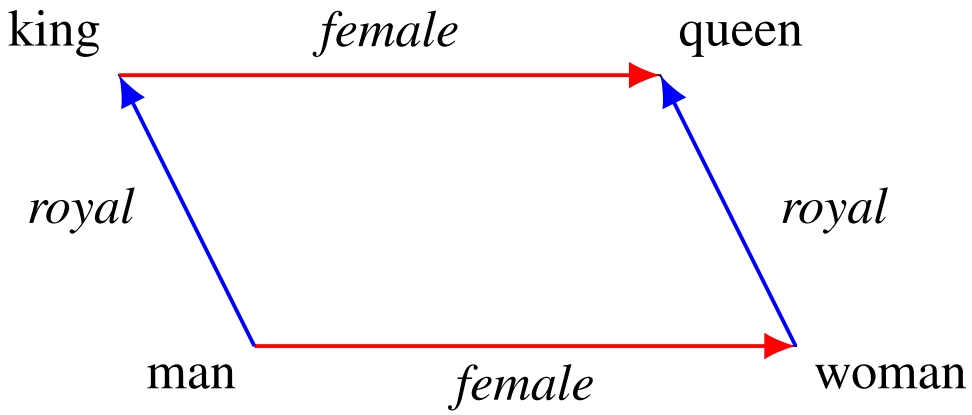
Window Size	Text	Skip-grams
2	[The **wide** road shimmered] in the hot sun.	wide, the wide, road wide, shimmered
	The [wide road **shimmered** in the] hot sun.	shimmered, wide shimmered, road shimmered, in shimmered, the
	The wide road shimmered in [the hot **sun**].	sun, the sun, hot
3	[The **wide** road shimmered in] the hot sun.	wide, the wide, road wide, shimmered wide, in
	[The wide road **shimmered** in the hot] sun.	shimmered, the shimmered, wide shimmered, road shimmered, in shimmered, the shimmered, hot
	The wide road shimmered [in the hot **sun**].	sun, in sun, the sun, hot

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$

Talk about negative sampling

Word analogies

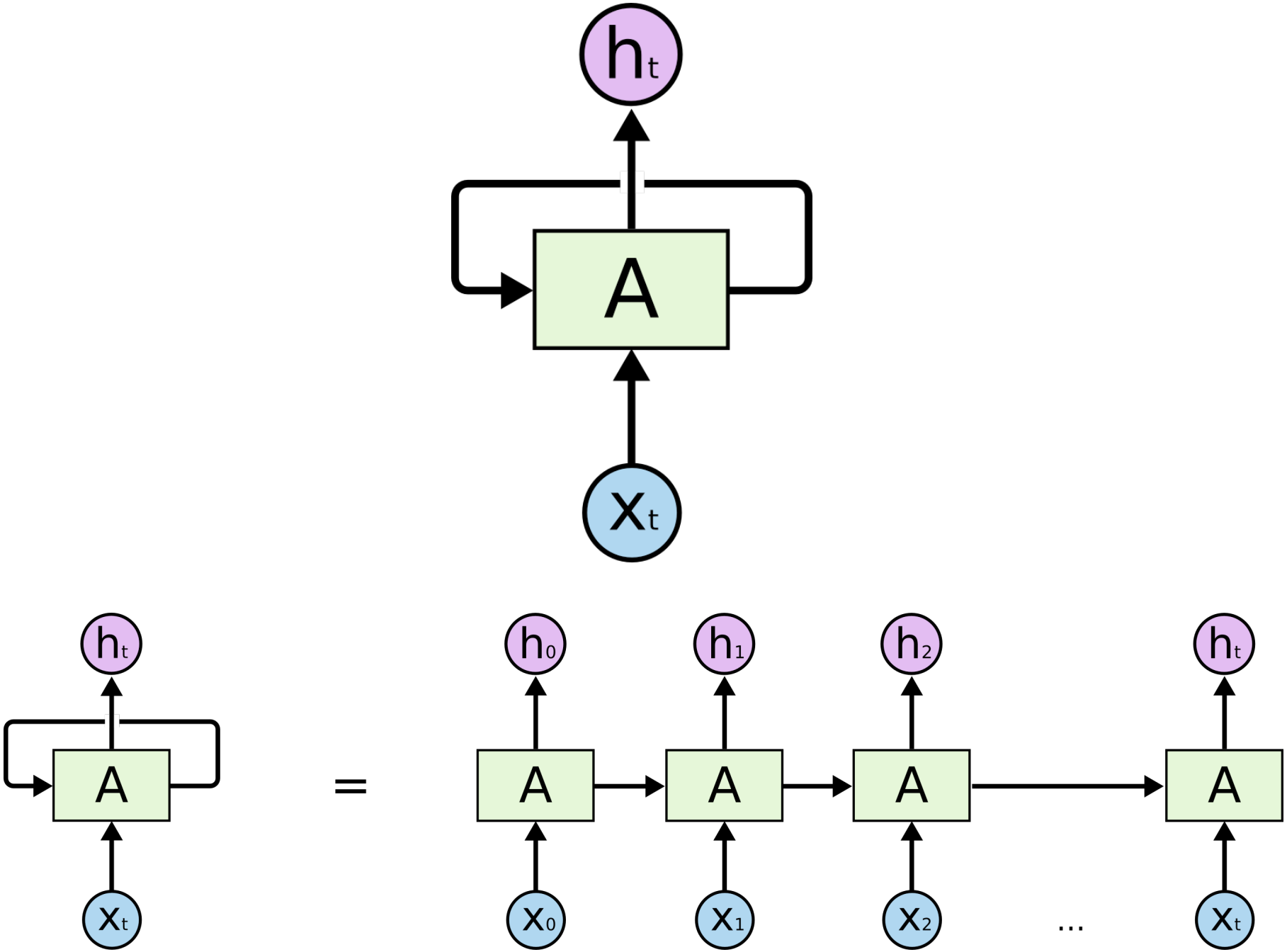


https://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html

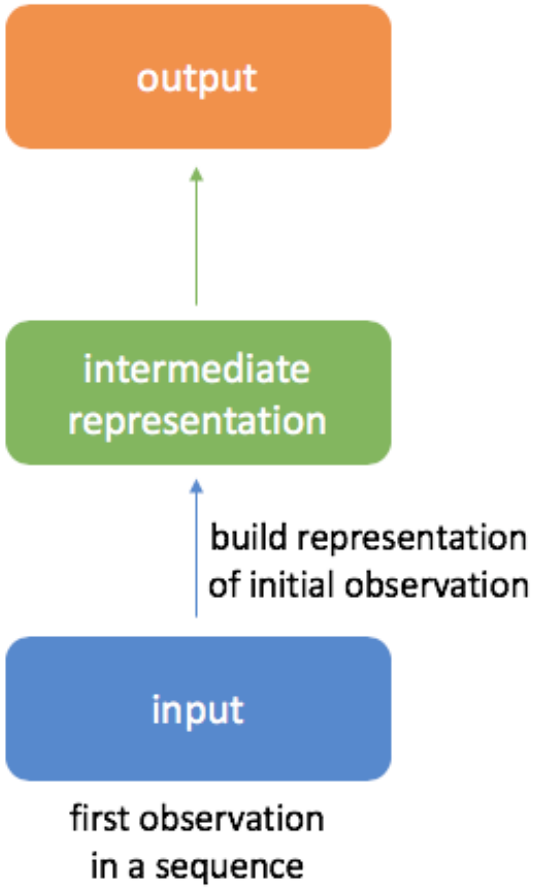
https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

Recurrent Neural Networks

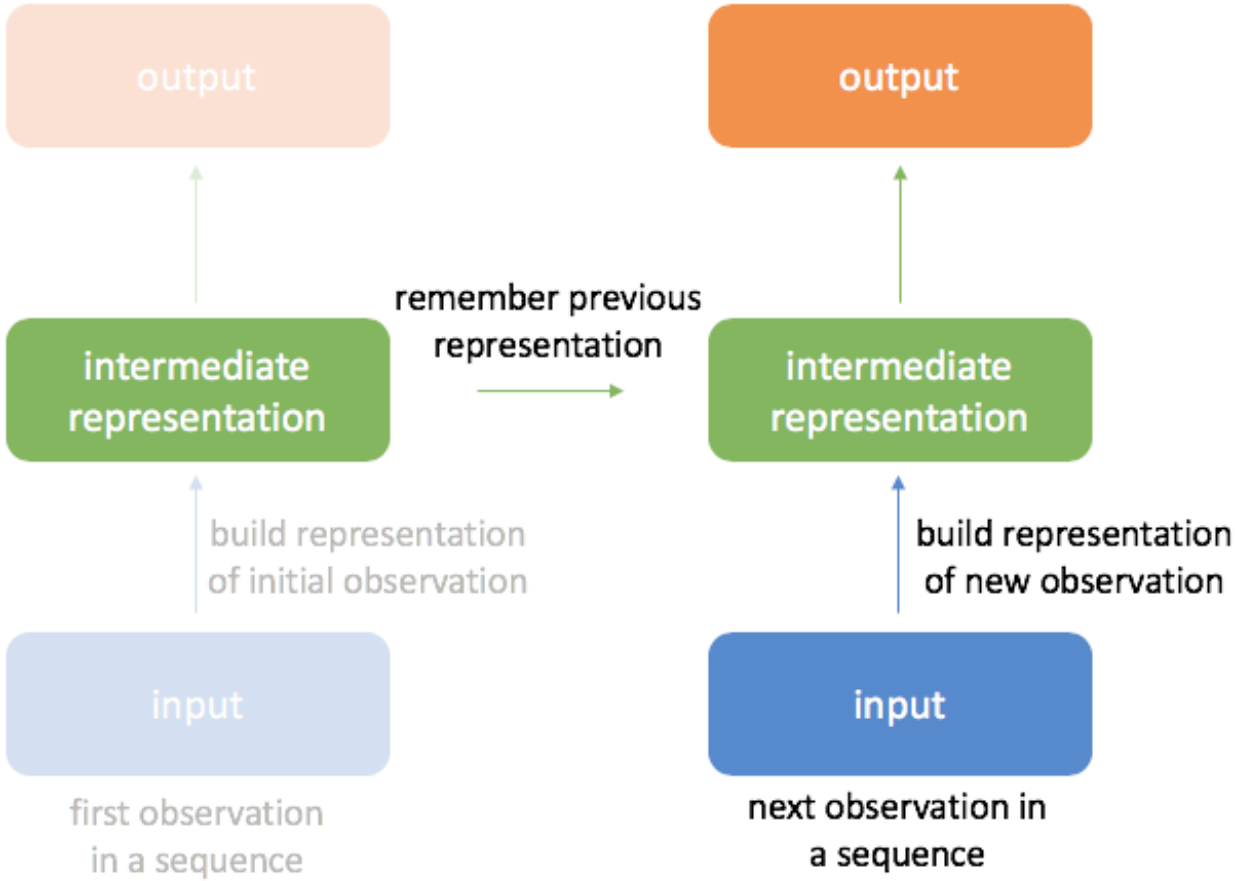
https://www.jeremyjordan.me/introduction-to-recurrent-neural-networks/

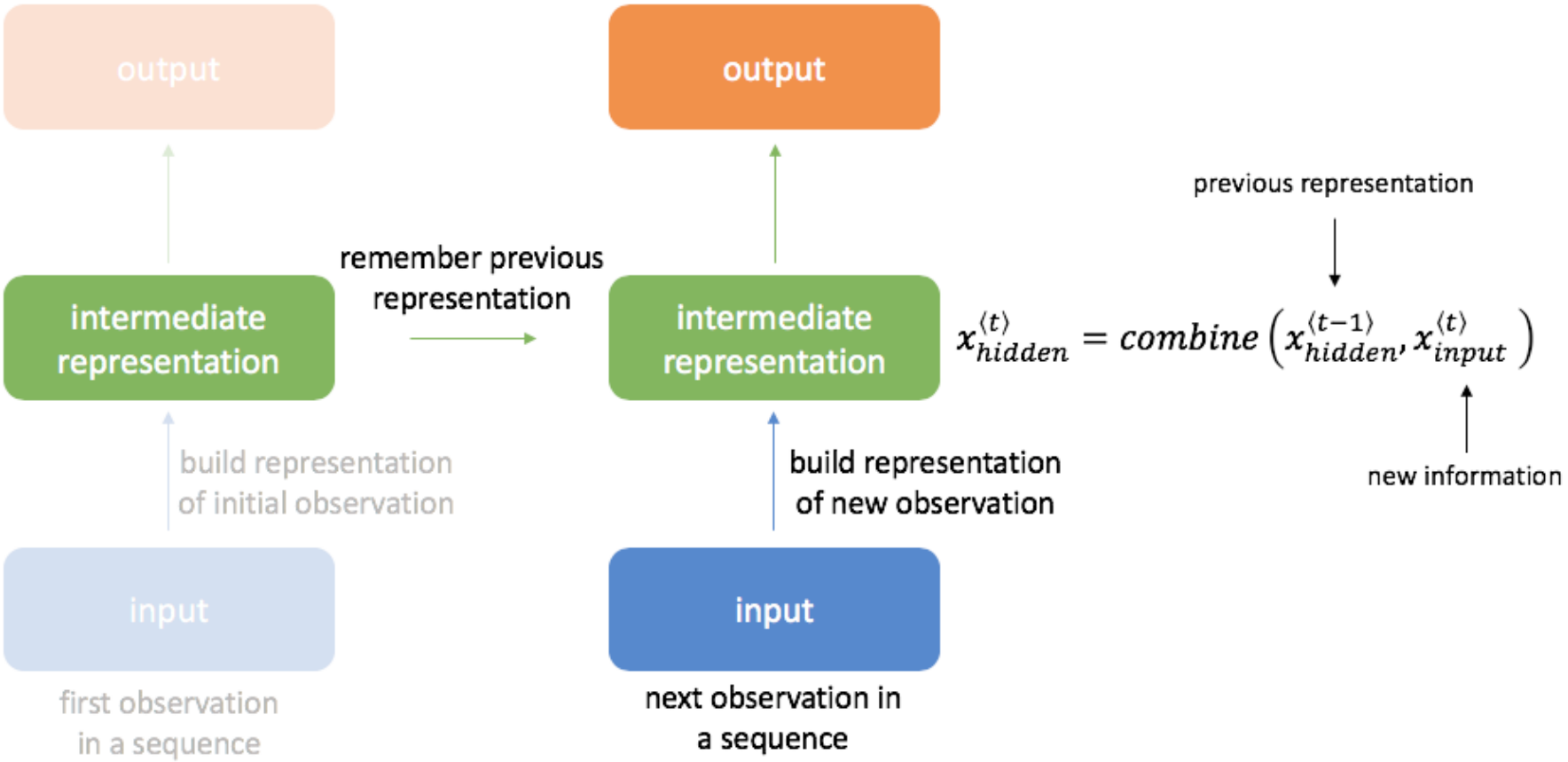

h_t
A
x_t
h_t
A
x_t
=
h_0
h_1
h_2
h_t
A
A
A
A
x_0
x_1
x_2
...
x_t

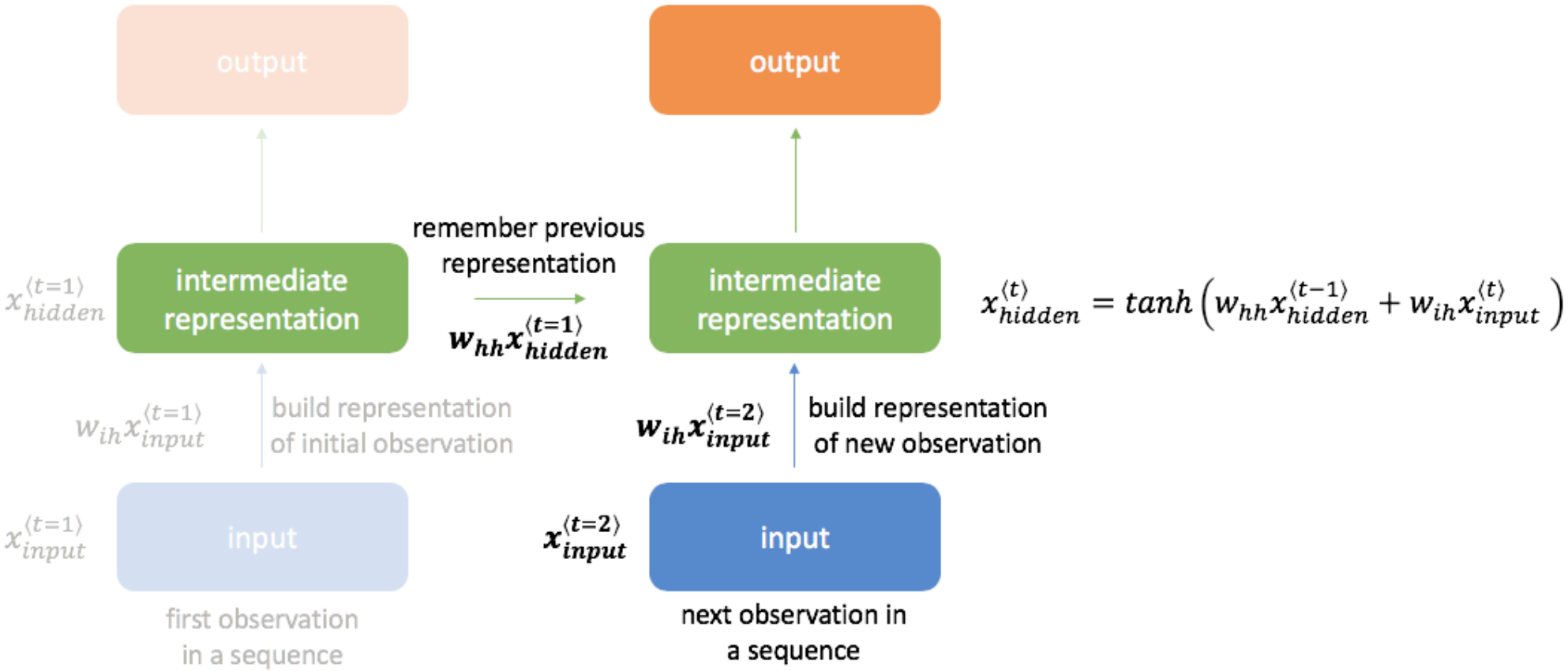
Non-sequential



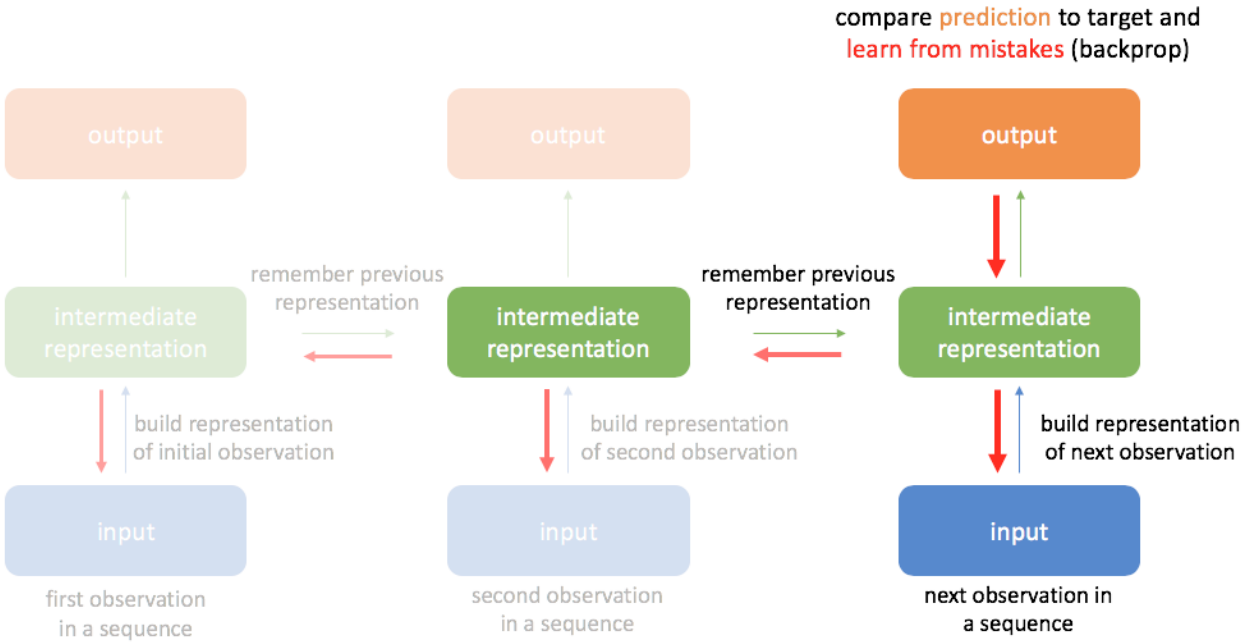
Sequential




output
output
previous representation
remember previous representation
intermediate representation
intermediate representation
$x_{hidden}^{\langle t \rangle} = combine\left(x_{hidden}^{\langle t-1 \rangle}, x_{input}^{\langle t \rangle}\right)$
build representation of initial observation
build representation of new observation
new information
input
input
first observation in a sequence
next observation in a sequence


output
output
remember previous representation
$x^{\langle t=1\rangle}_{hidden}$
intermediate representation
intermediate representation
$w_{hh}x^{\langle t=1\rangle}_{hidden}$
$x^{\langle t\rangle}_{hidden} = tanh\left(w_{hh}x^{\langle t-1\rangle}_{hidden} + w_{ih}x^{\langle t\rangle}_{input}\right)$
$w_{ih}x^{\langle t=1\rangle}_{input}$
build representation of initial observation
$w_{ih}x^{\langle t=2\rangle}_{input}$
build representation of new observation
$x^{\langle t=1\rangle}_{input}$
input
$x^{\langle t=2\rangle}_{input}$
input
first observation in a sequence
next observation in a sequence

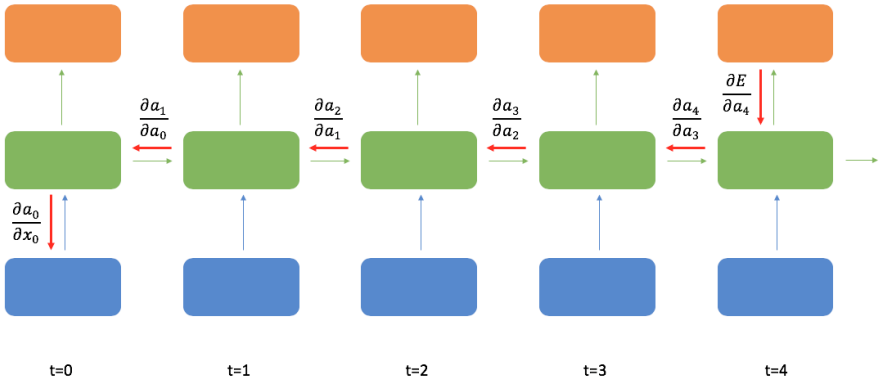
Backprop



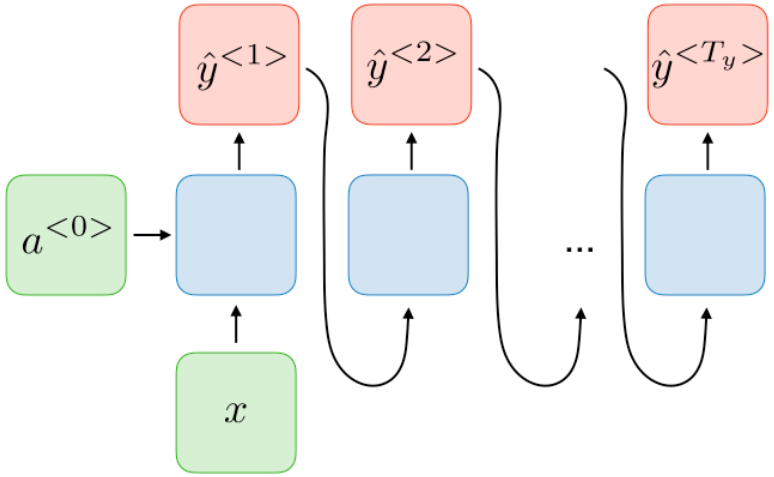
Notation:

$a_t = x^{(t=t)}_{hidden}$ (the output of a recurrent layer)

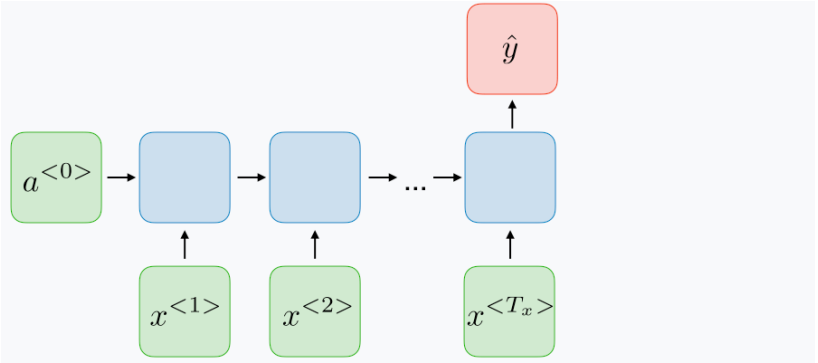
$x_t = x^{(t=t)}_{input}$ (the input to a recurrent layer)



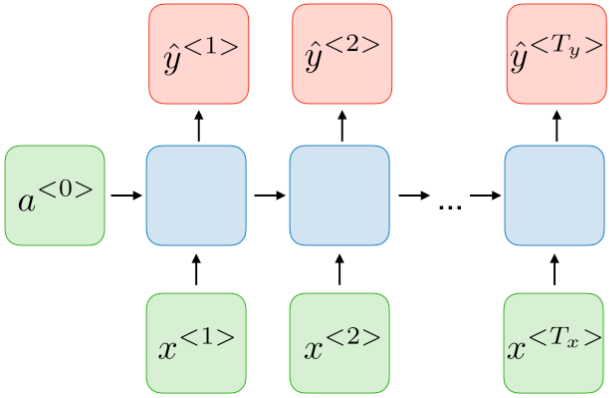
One-to-many



Many-to-one



Many-to-Many

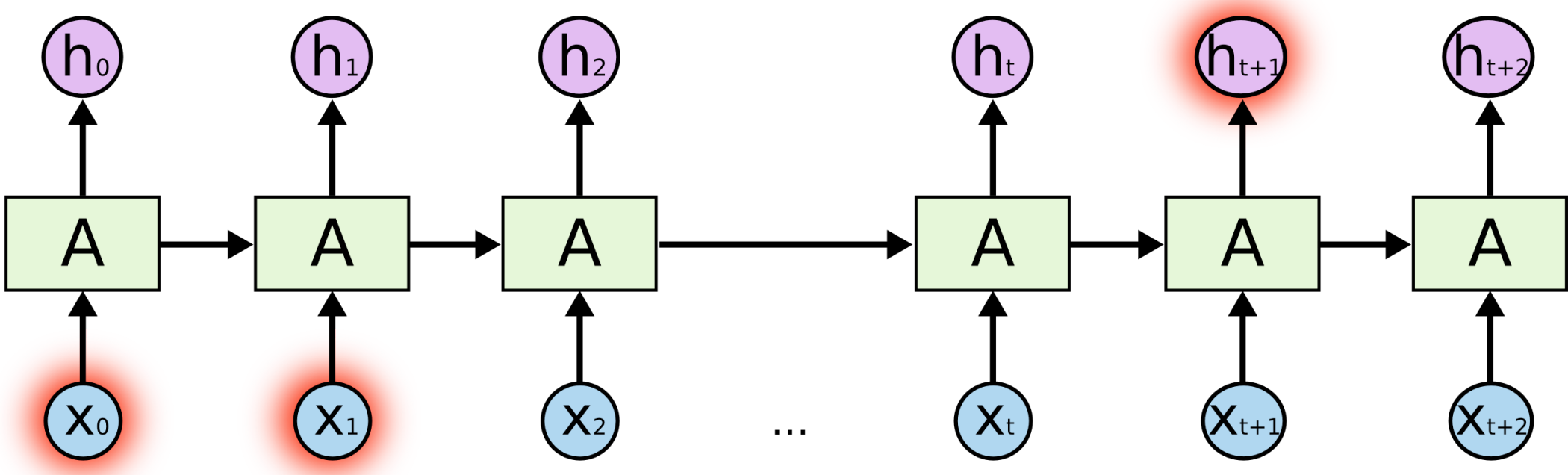


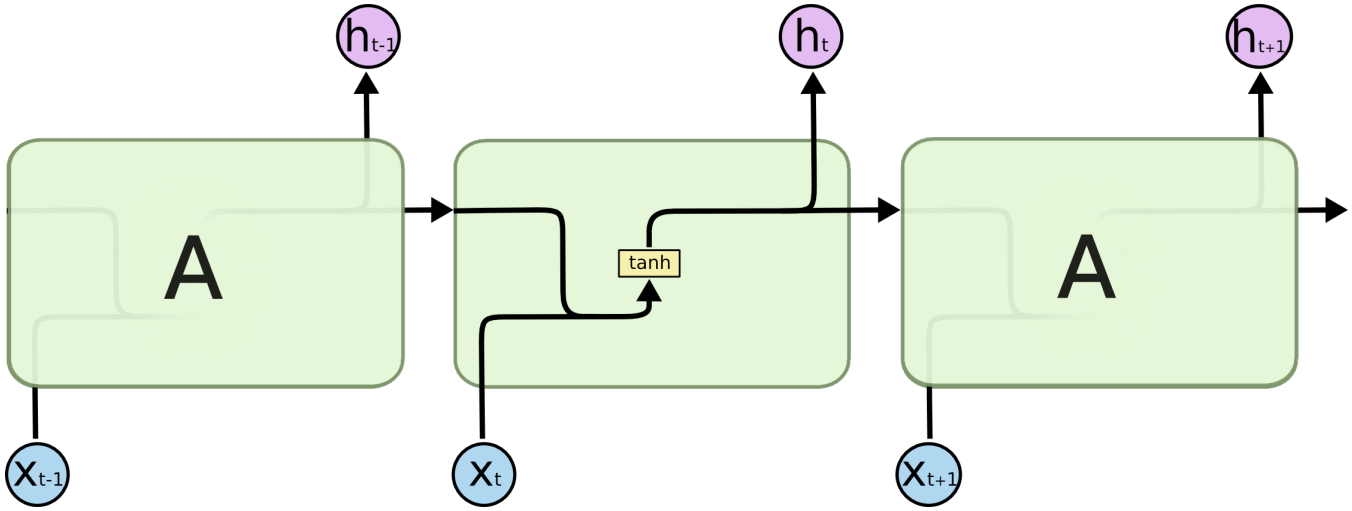
LTSM- Networks

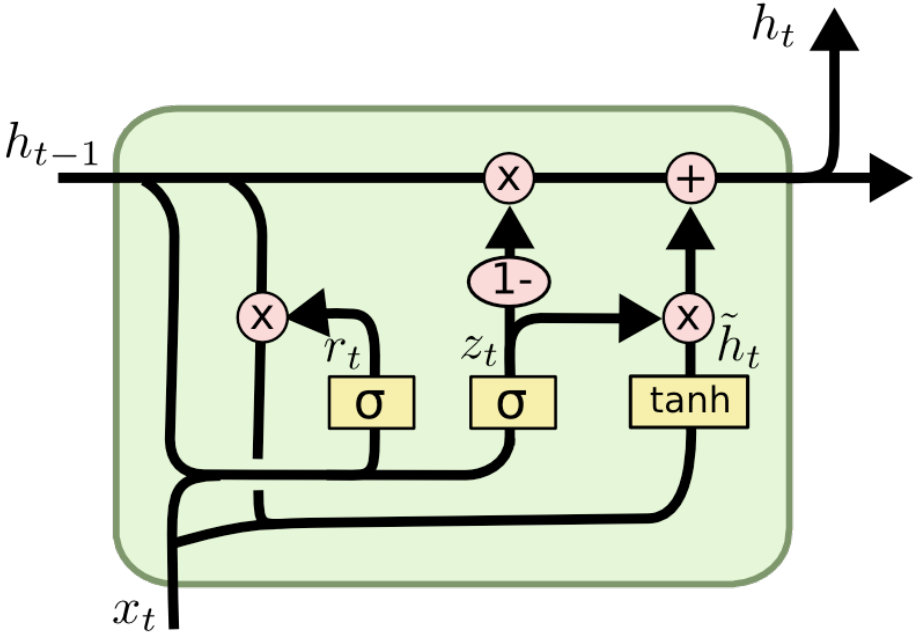
https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

How do you remember things for long times?
Many layers….







$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

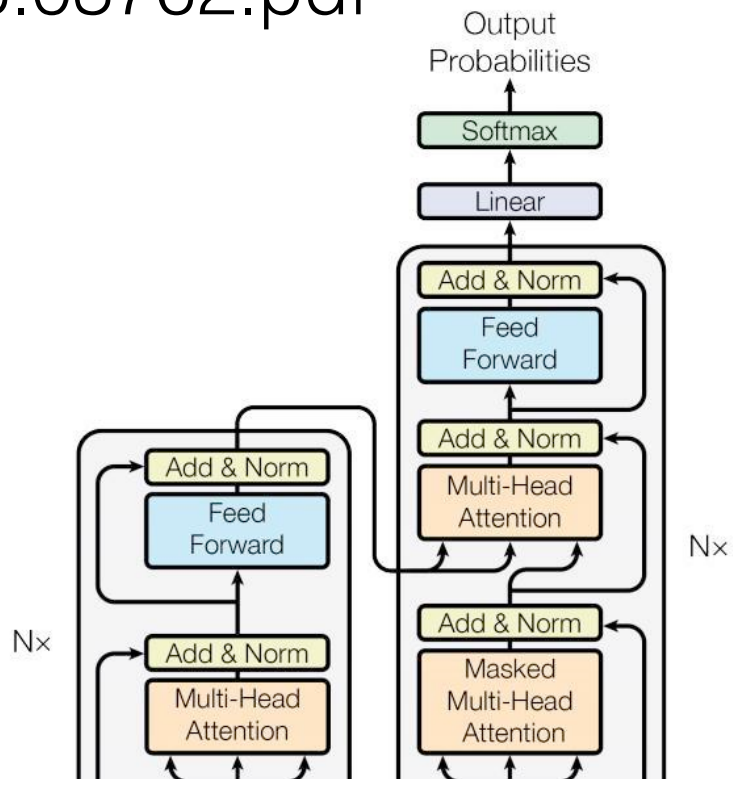
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Transformers

https://pytorch.org/tutorials/beginner/transformer_tutorial.html

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



Tranformers: Language Translation