A shallow introduction to deep learning for agents

Jerry Spanakis & Kurt Driessens

Department of **Data Science** and **Knowledge Engineering**
Maastricht University

Maastricht University

Tutorial Overview

- Part 1:
  - From linear and logistic regression to neural networks
  - Autoencoders
  - Convolutional Networks
  - Deep Network Learning
  - Applied to Agents

!You need part 1 for any deep learning architecture!

- Part 2:
  - Introduction to Recurrent Neural Networks (RNNs)
  - RNN variants
  - Generative Adversarial Networks
  - Summary
  - Demo?

INPUTS/
OUTPUTS

ARCHITECTURE

LOSS

Introduction to RNNs

Jerry Spanakis

Some slide credits (esp. illustrations): Arun Mallya (github link)

Motivation

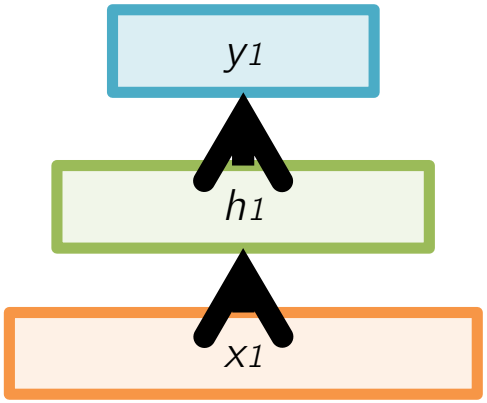
- Not all problems can be converted into one with fixed-length inputs and outputs
- Problems such as Speech Recognition or Time-series Prediction require a system to store and use context information
  - Simple case: Output YES if the number of 1s is even, else NO
    1000010101 – YES, 100011 – NO, …
- Hard/Impossible to choose a fixed context window
  - There can always be a new sample longer than anything seen

Recurrent Neural Networks (RNNs)

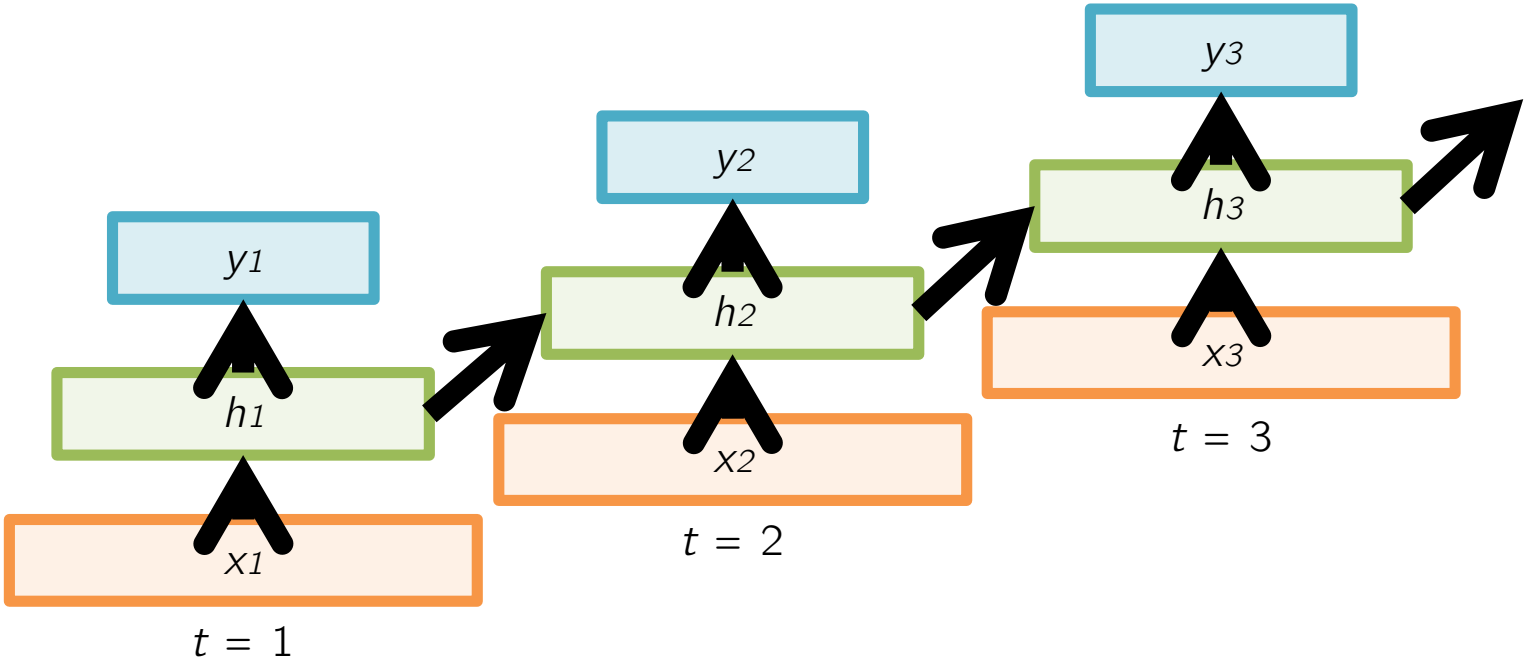
- **R**ecurrent **N**eural **N**etwork**s** take the previous output or hidden states as inputs.
  The composite input at time t has some historical information about the happenings at time T < t

- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori

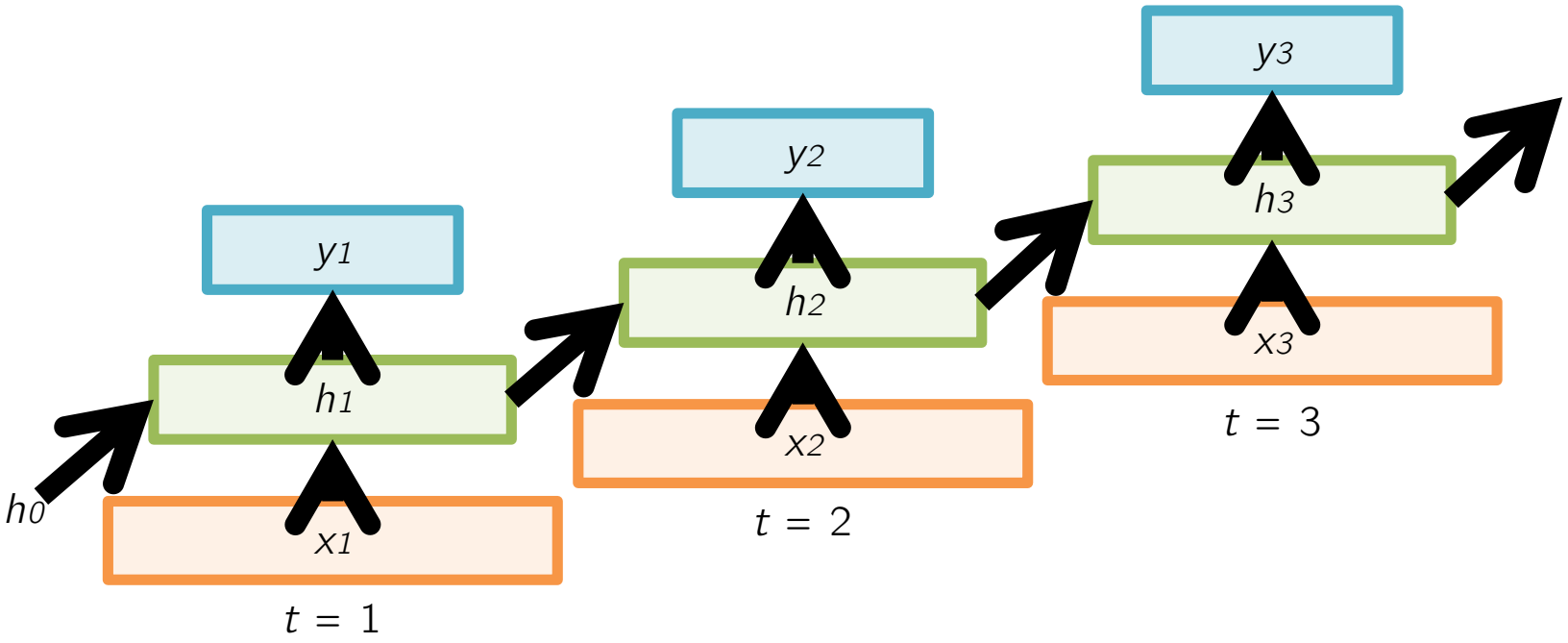
Sample Feed-forward Network



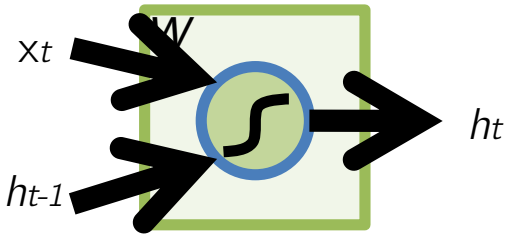
Sample RNN



Sample RNN

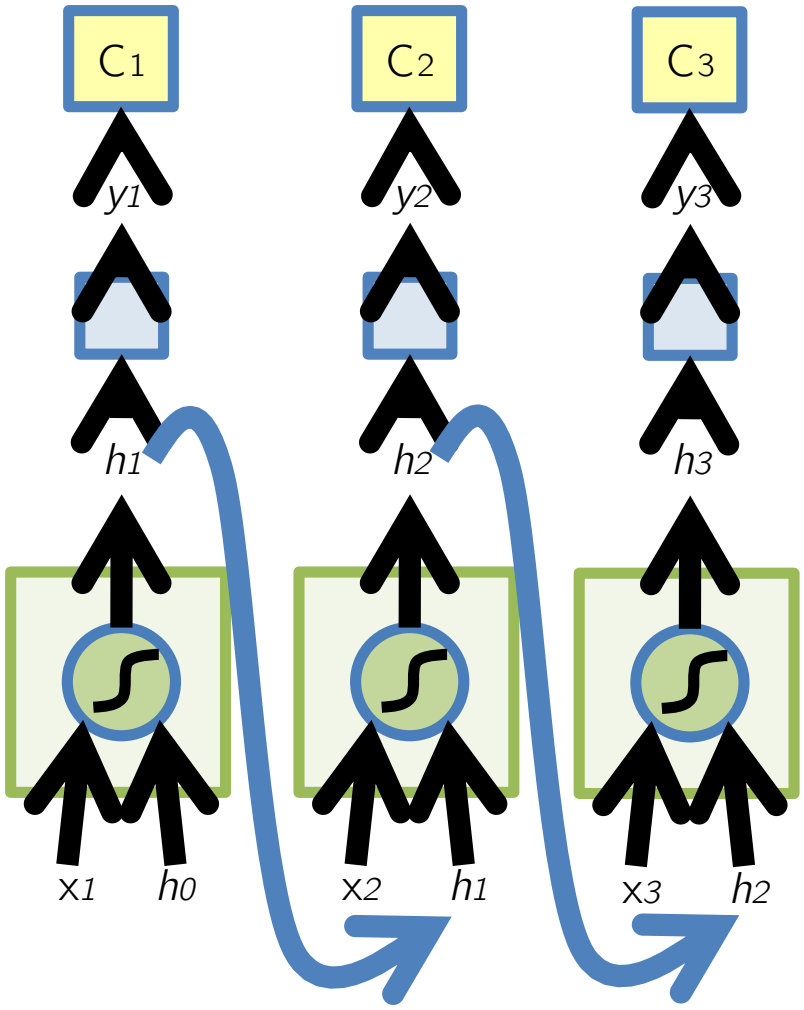


The Vanilla RNN Cell



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

The Vanilla RNN Forward

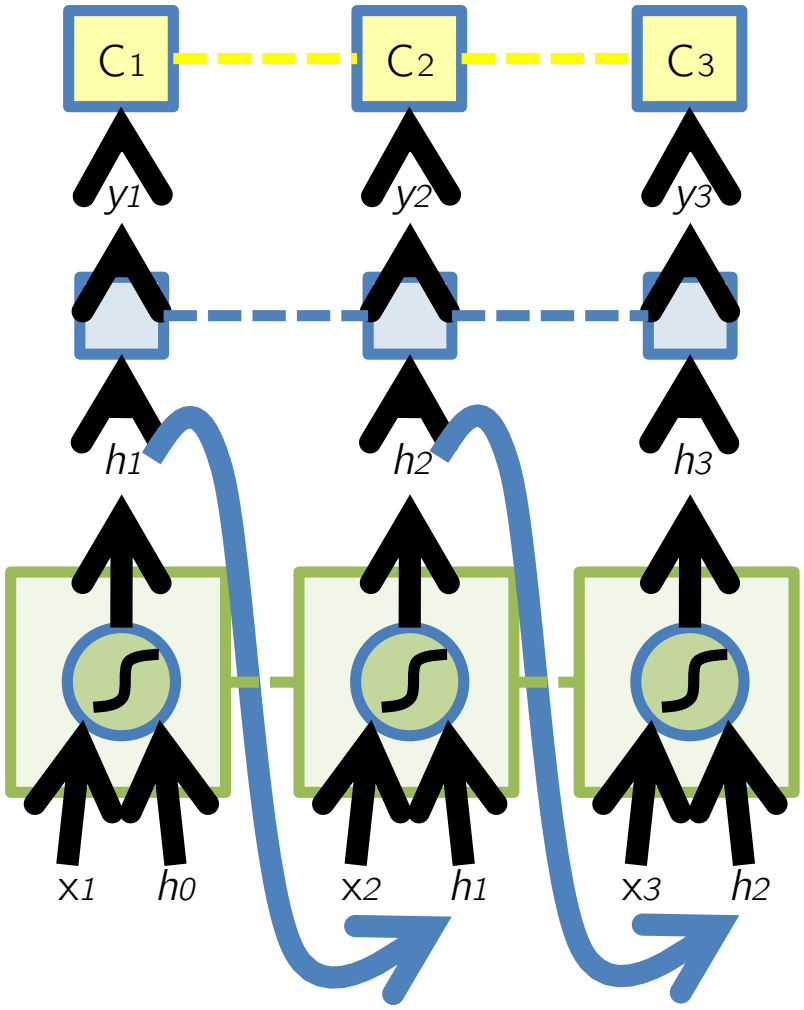


$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

The Vanilla RNN Forward



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

indicates shared weights

Recurrent Neural Networks (RNNs)

- Note that the weights are shared over time

- Essentially, copies of the RNN cell are made over time (unrolling/unfolding), with different inputs at different time steps

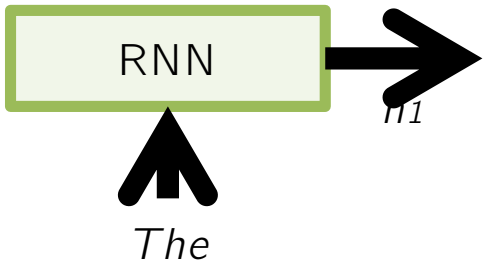
Sentiment Classification

- Classify a
  …restaurant review from Yelp!
  …movie review from IMDB
  …
  as positive or negative

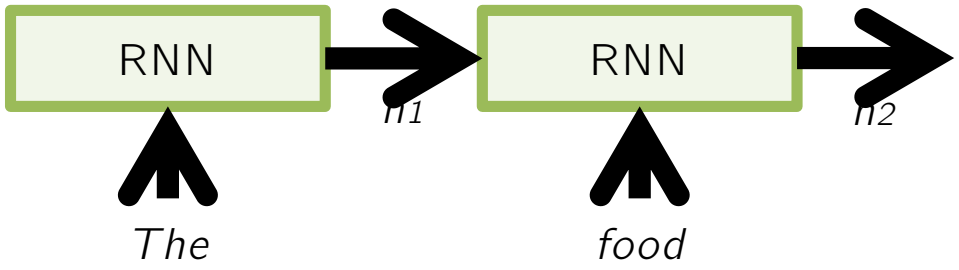
- **Inputs:** Multiple words, one or more sentences
- **Outputs:** Positive / Negative classification

- "The food was really good"
- "The chicken crossed the road because it was uncooked"

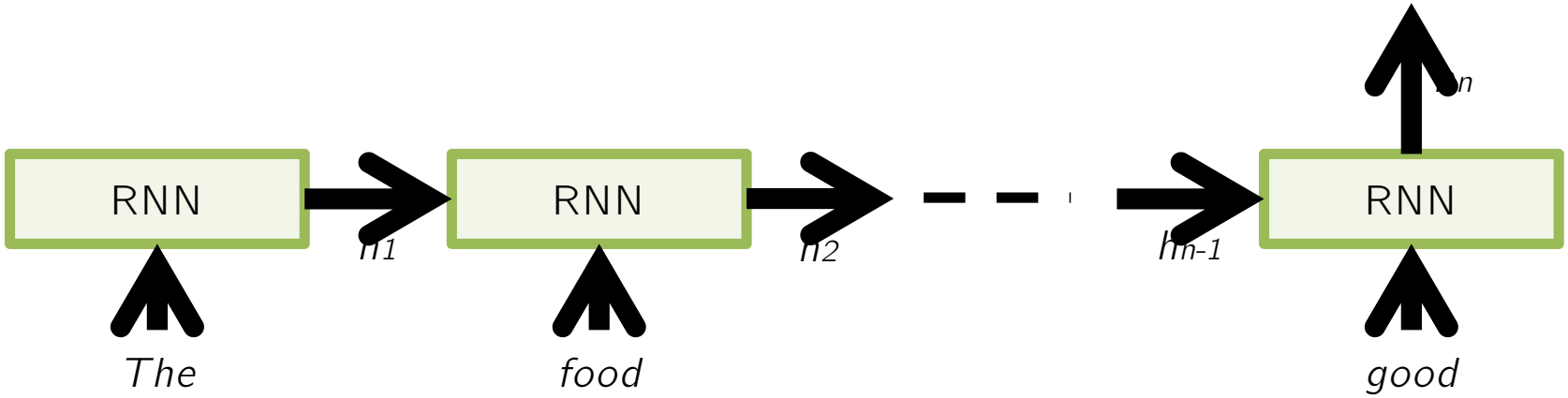
Sentiment Classification



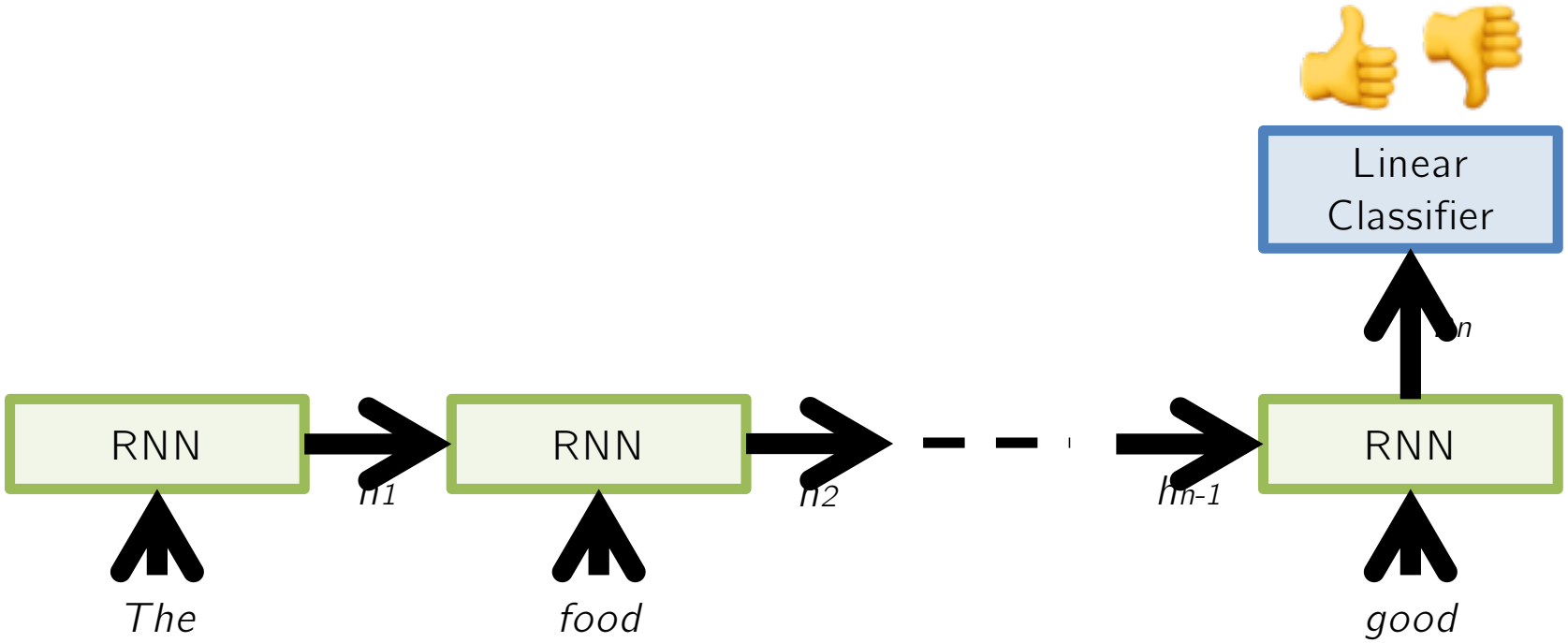
Sentiment Classification



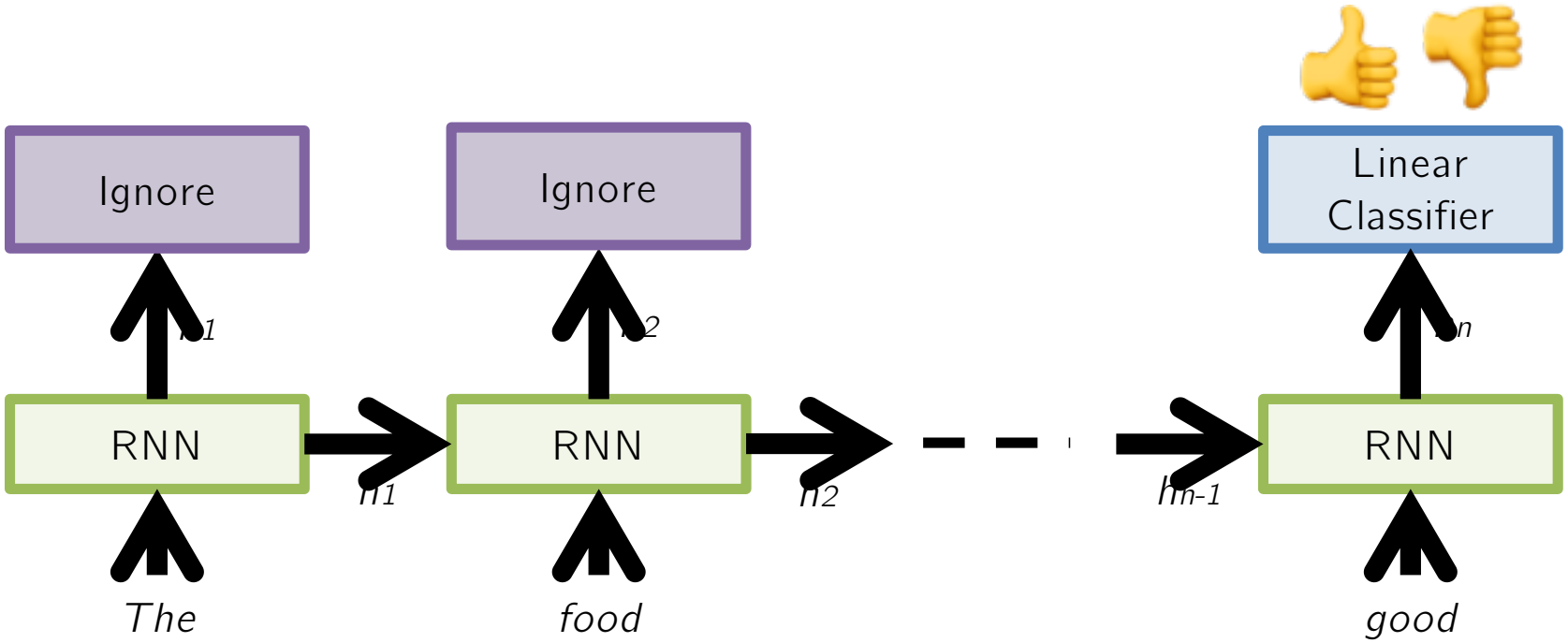
Sentiment Classification



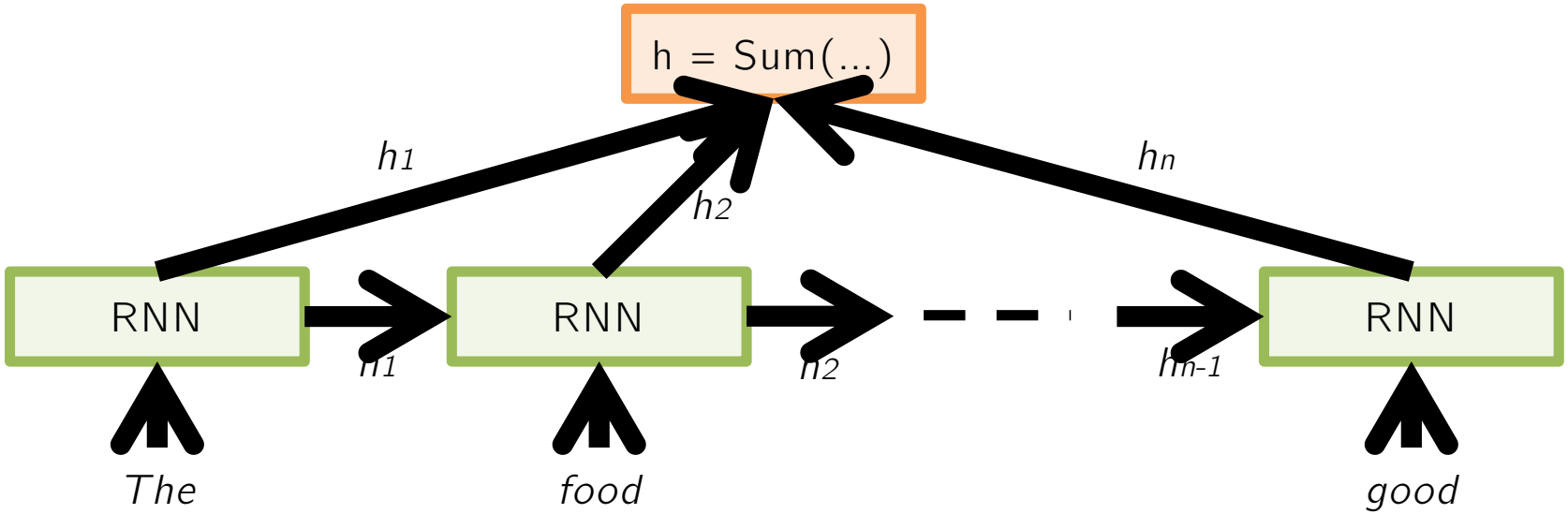
Sentiment Classification



Sentiment Classification



Sentiment Classification (alt.)



Sentiment Classification (alt.)

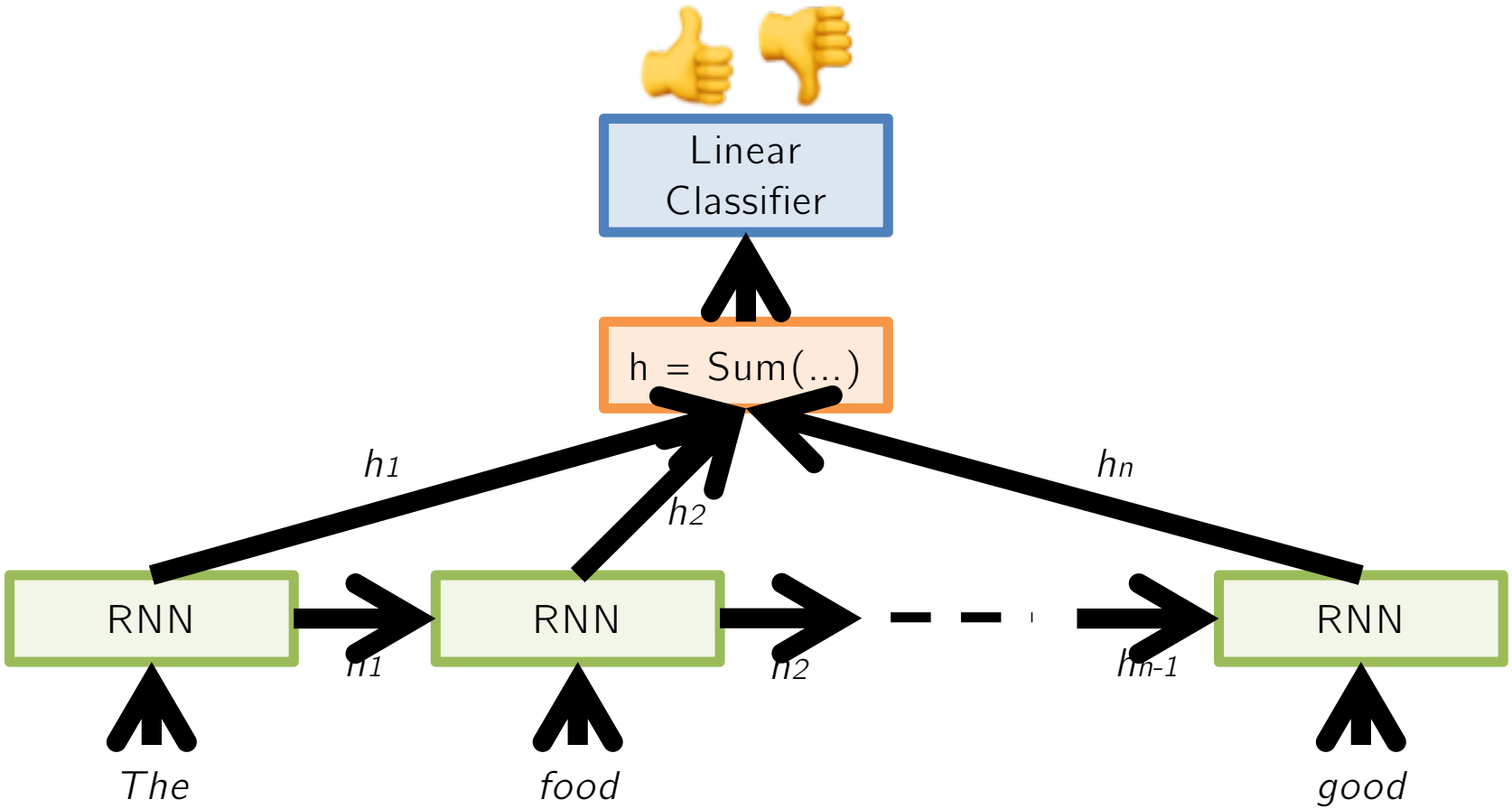


Image Captioning

- Given an image, produce a sentence describing its contents

- **Inputs:** Image feature (from a CNN)
- **Outputs:** Multiple words (let's consider one sentence)

→ The dog is hiding

Image Captioning

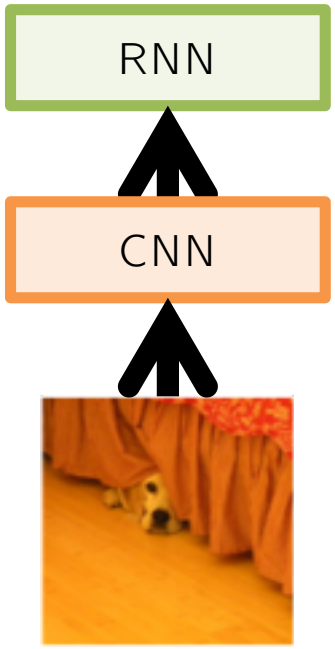


Image Captioning

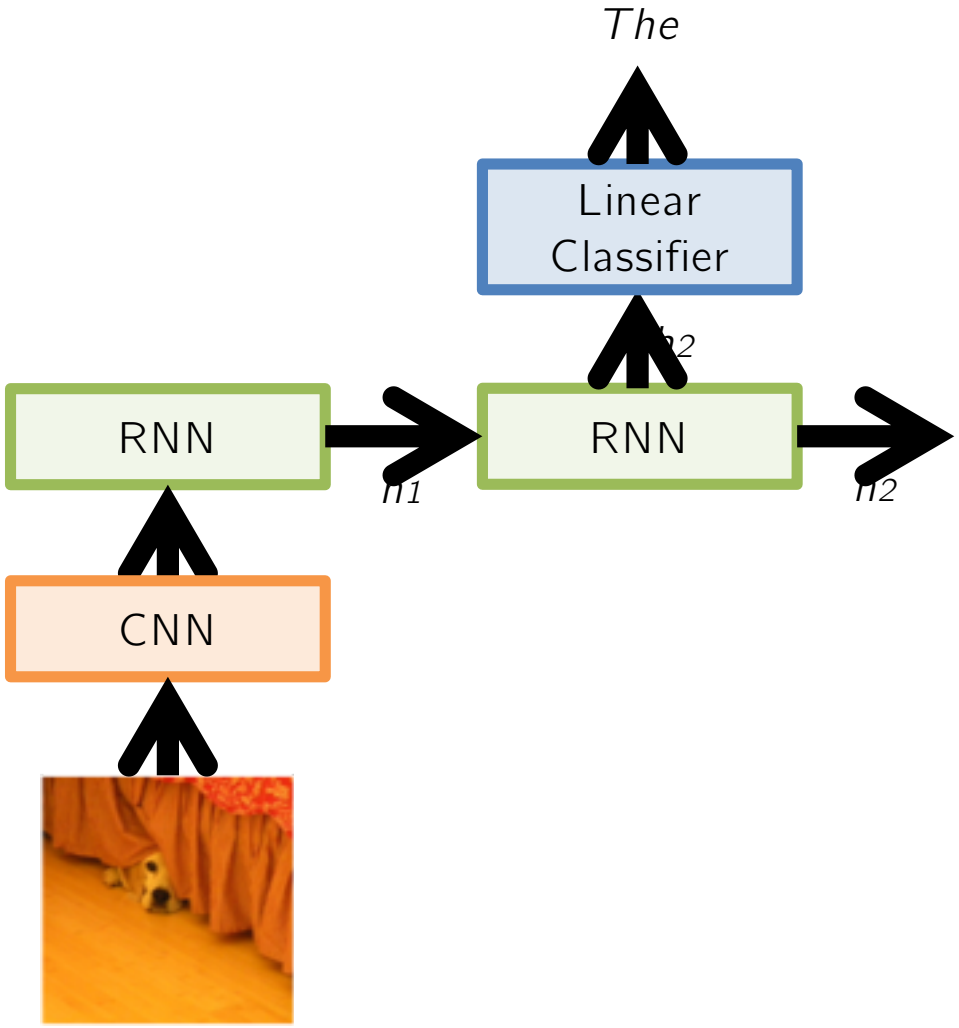
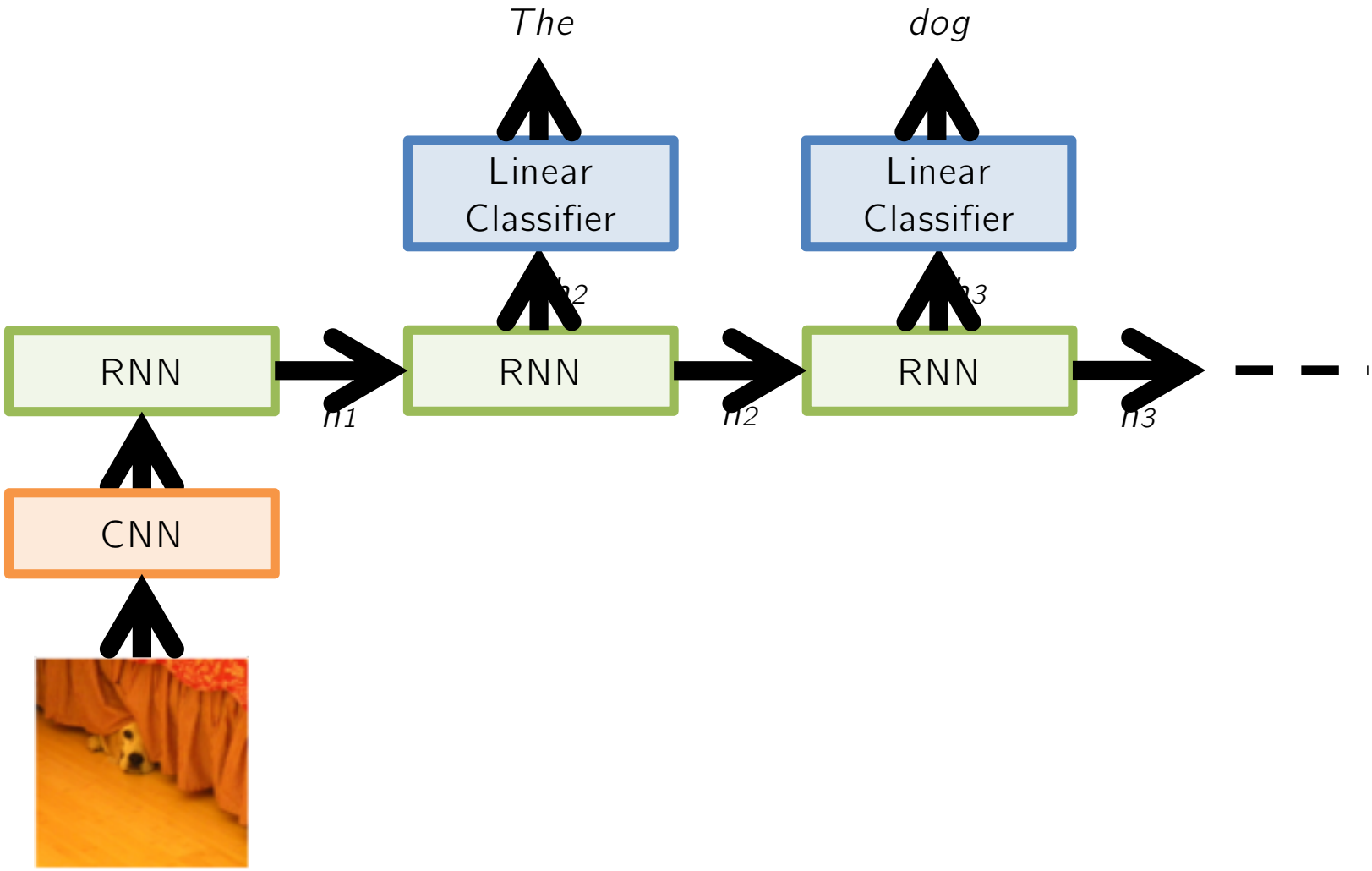


Image Captioning



RNN Outputs: Image Captions

Show and Tell: A Neural Image Caption Generator, CVPR 15

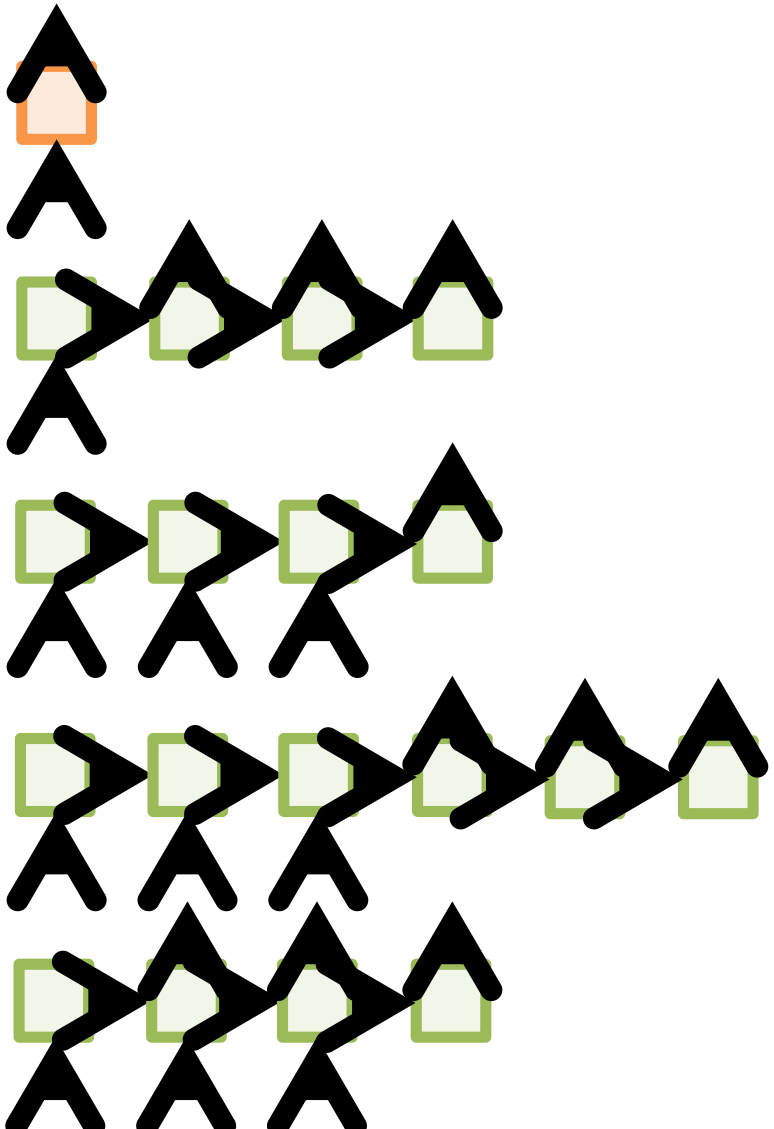
RNN Outputs: Language Modeling

VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Input – Output Scenarios



Single - Single

Feed-forward Network

Single - Multiple

Image Captioning

Multiple - Single

Sentiment Classification

Multiple - Multiple

Translation

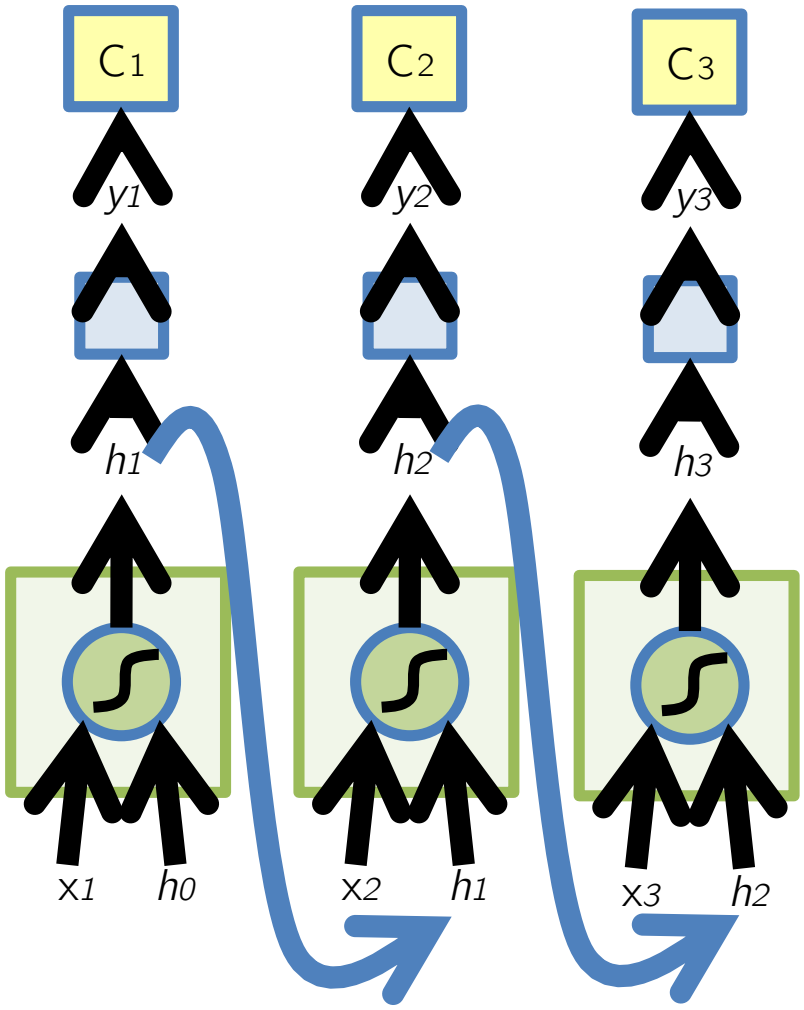
Image Captioning (again...?)

Input – Output Scenarios

Note: We might deliberately choose to frame our problem as a particular input-output scenario for ease of training or better performance.

For example, at each time step, provide previous word as input for image captioning
(Single-Multiple to Multiple-Multiple).

The Vanilla RNN Forward



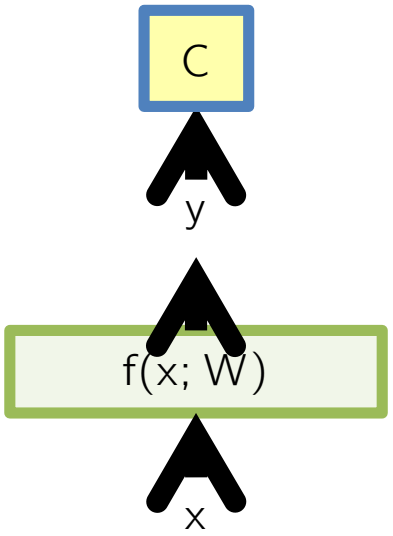
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

"Unfold" network through time by making copies at each time-step

Remember BackPropagation?



$$y = f(x; W)$$

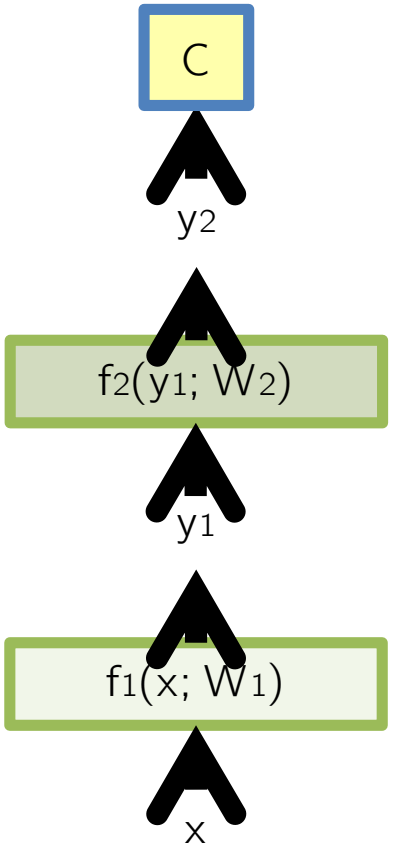
$$C = \text{Loss}(y, y_{GT})$$

SGD Update

$$W \leftarrow W - \eta \frac{\partial C}{\partial W}$$

$$\frac{\partial C}{\partial W} = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right)$$

Reminder: Multiple Layers



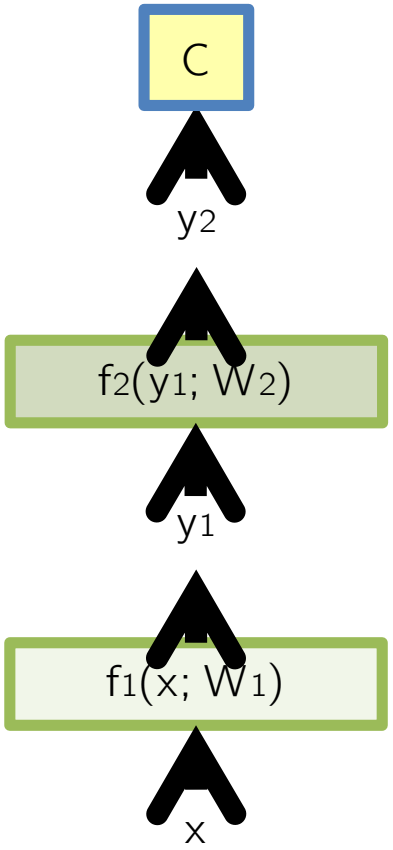
$$y_1 = f_1(x; W_1)$$
$$y_2 = f_2(y_1; W_2)$$
$$C = \text{Loss}(y_2, y_{GT})$$

SGD Update

$$W_2 \leftarrow W_2 - \eta \frac{\partial C}{\partial W_2}$$

$$W_1 \leftarrow W_1 - \eta \frac{\partial C}{\partial W_1}$$

Reminder: Chain Rule for Gradient



$$y_1 = f_1(x; W_1)$$

$$y_2 = f_2(y_1; W_2)$$

$$C = \text{Loss}(y_2, y_{GT})$$

$$\text{Find } \frac{\partial C}{\partial W_1}, \frac{\partial C}{\partial W_2}$$

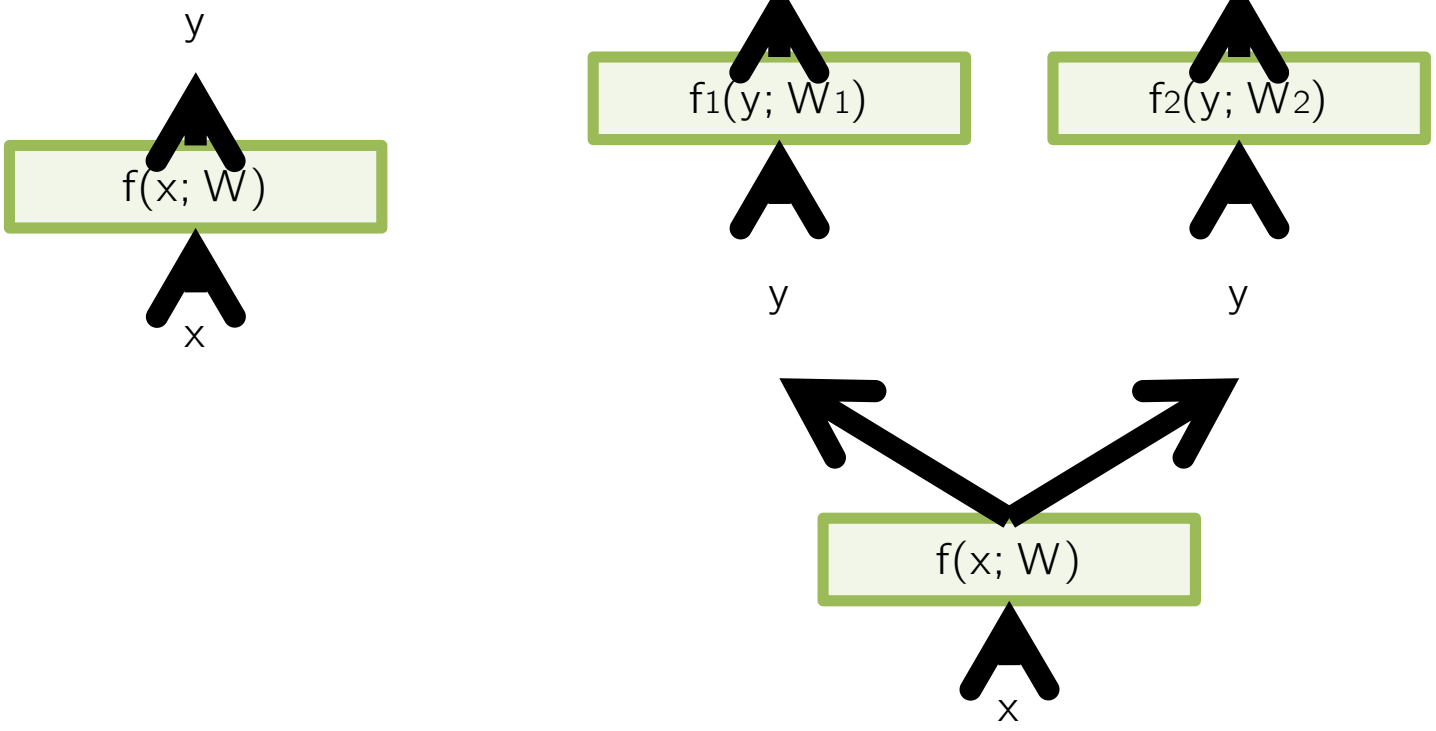
$$\frac{\partial C}{\partial W_2} = \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial W_2}\right)$$

$$\frac{\partial C}{\partial W_1} = \left(\frac{\partial C}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

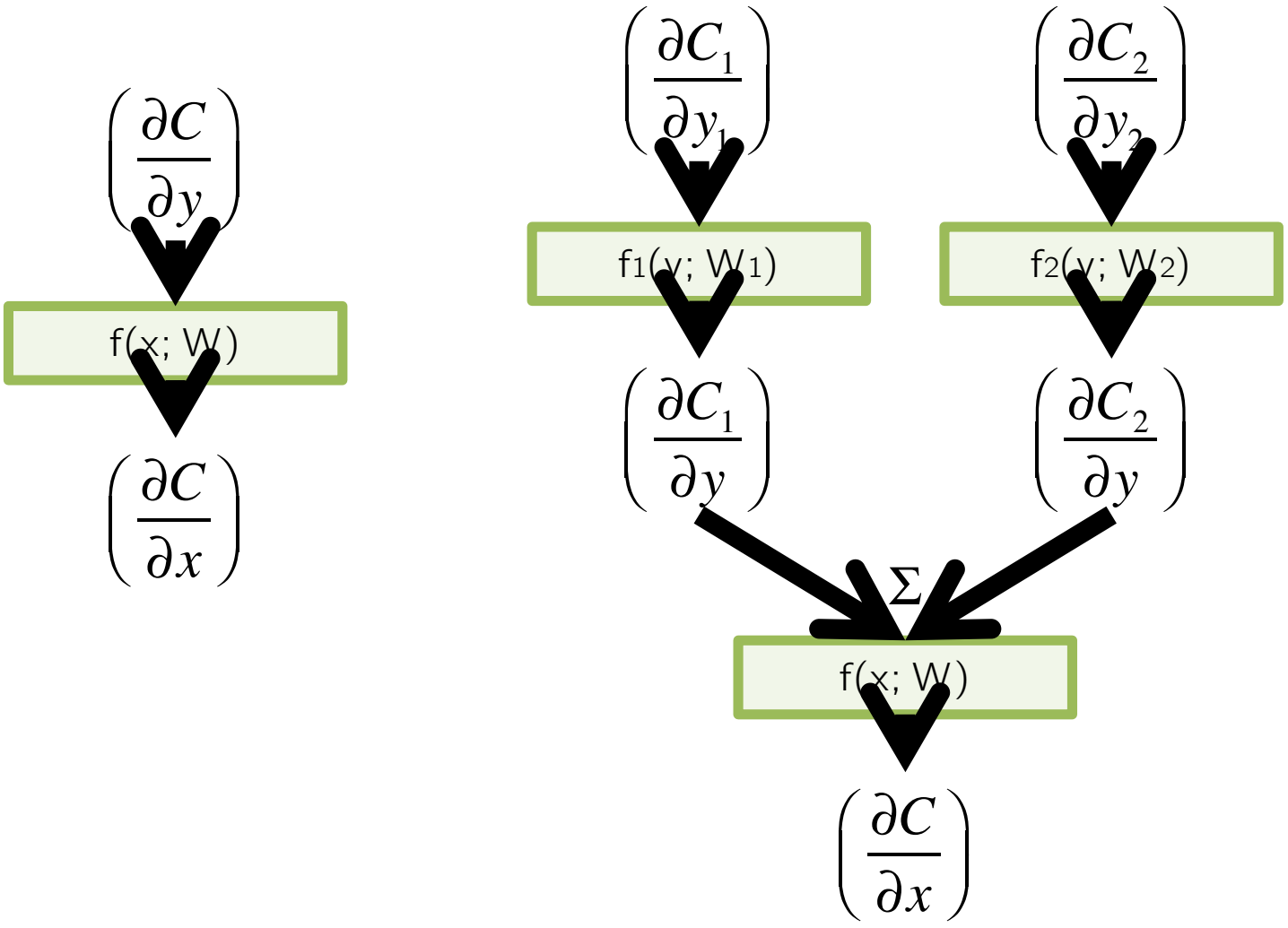
$$= \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

Application of the Chain Rule

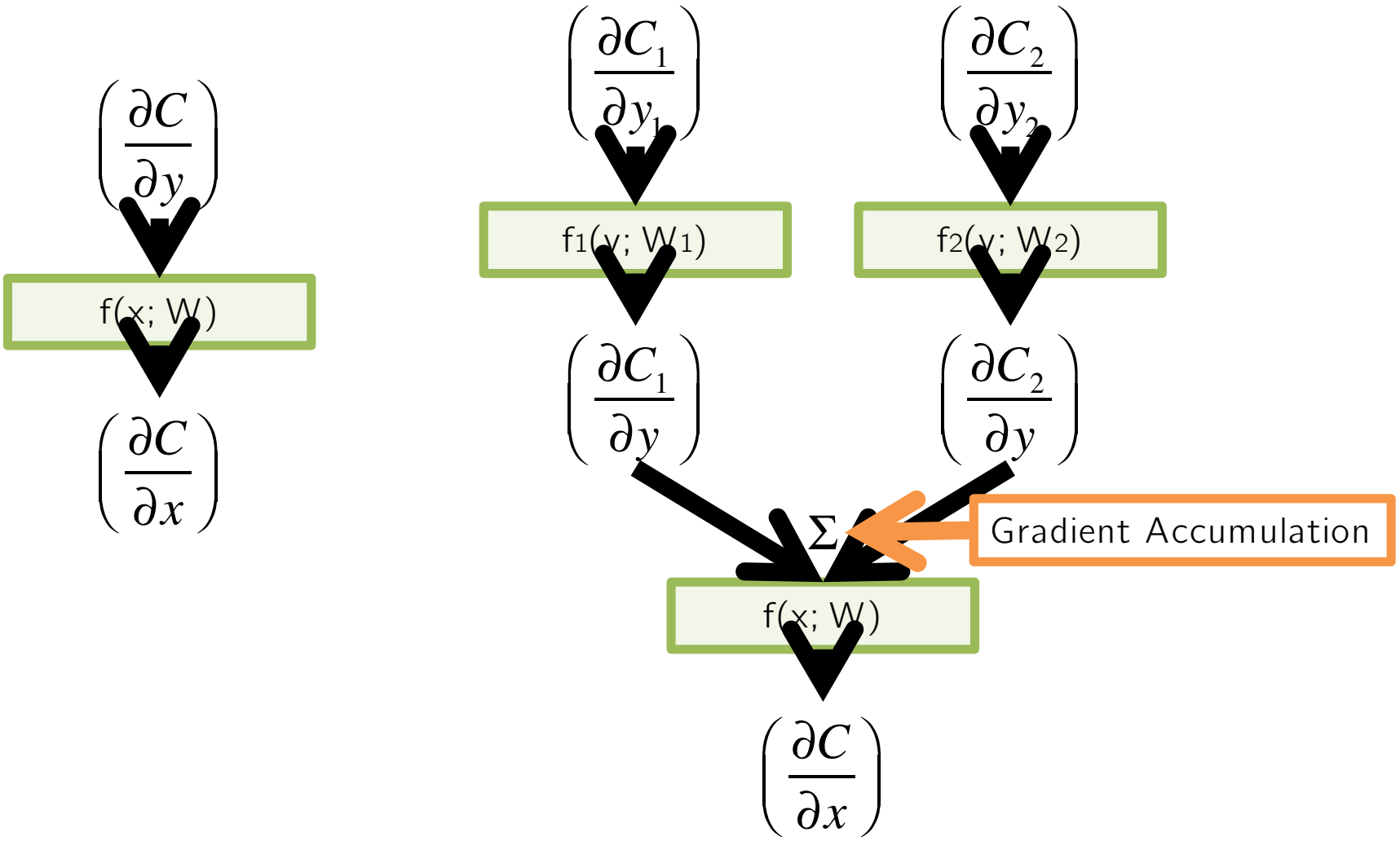
Extension to Computational Graphs



Extension to Computational Graphs



Extension to Computational Graphs

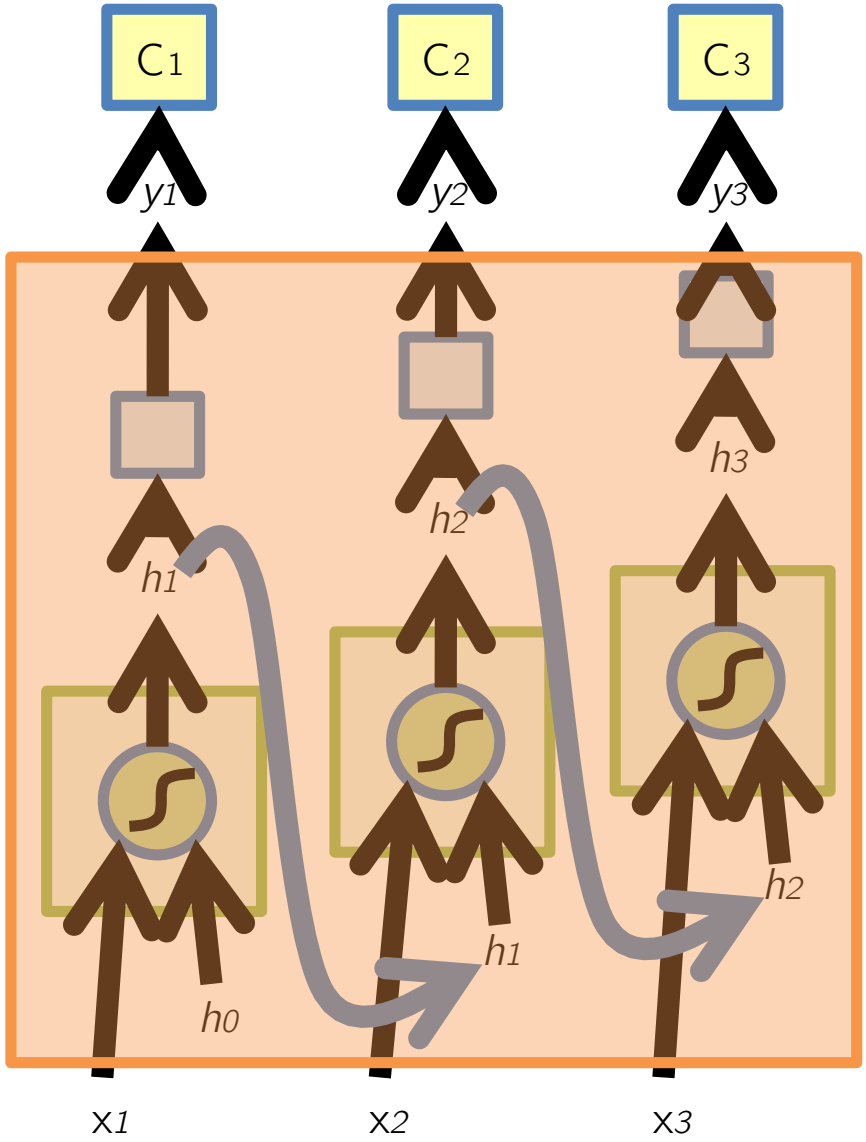


BackPropagation Through Time (BPTT)

- One of the methods used to train RNNs
- The unfolded network (used during forward pass) is treated as one big feed-forward network
- This unfolded network accepts the whole time series as input

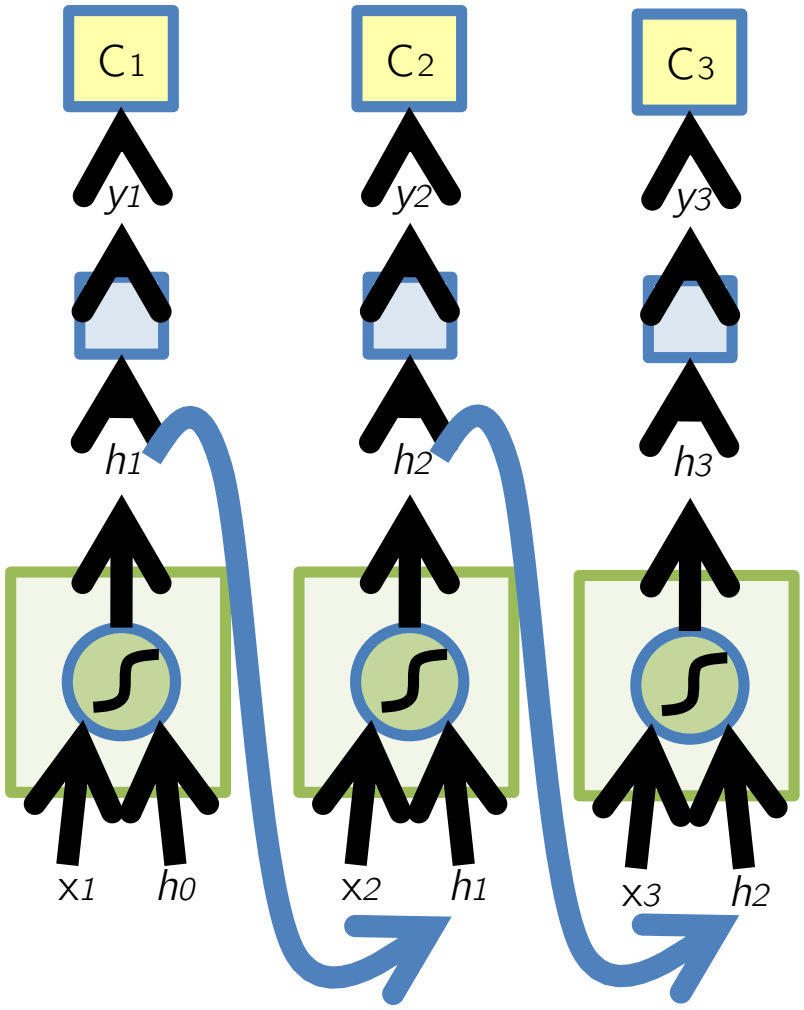
- The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and then applied to the RNN weights

The Unfolded Vanilla RNN

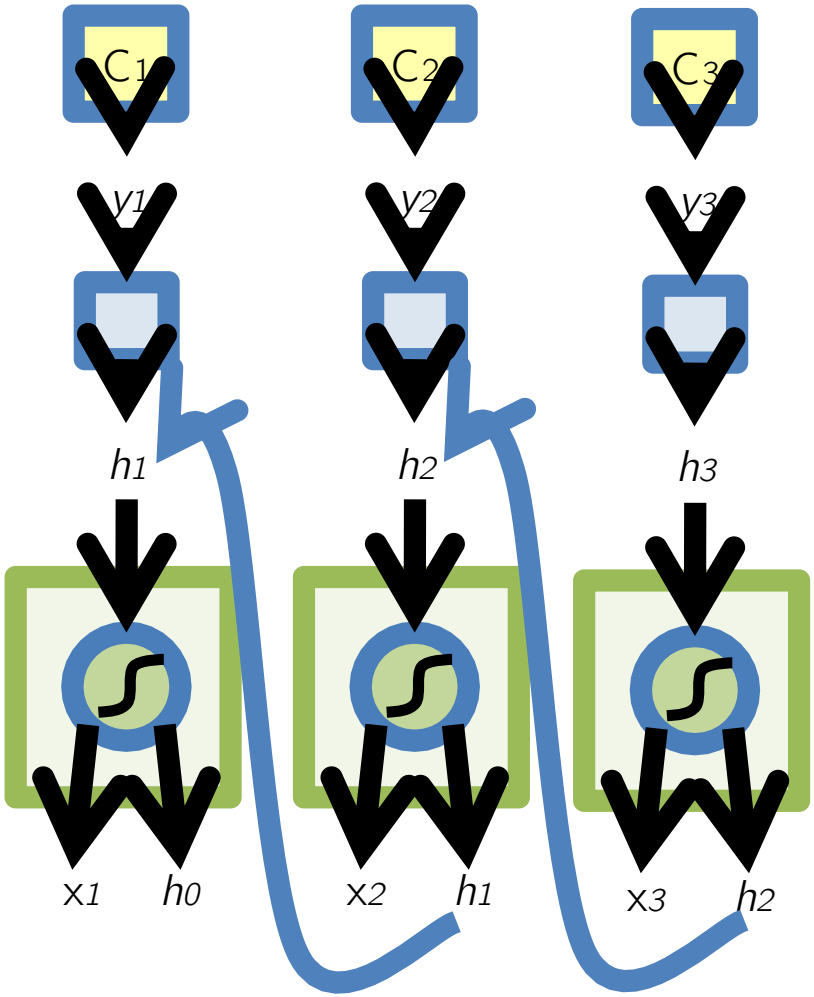


- Treat the unfolded network as one big feed-forward network!
- This big network takes in entire sequence as an input
- Compute gradients through the usual backpropagation
- Update shared weights

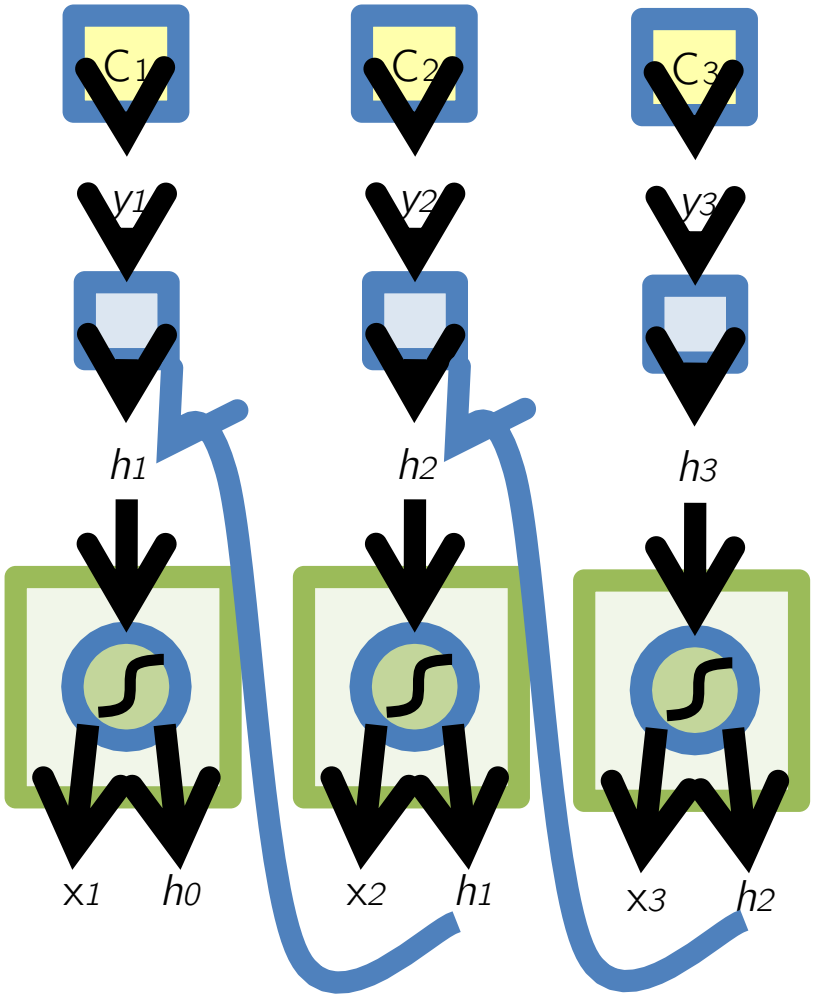
The Unfolded Vanilla RNN Forward



The Unfolded Vanilla RNN Backward



The Vanilla RNN Backward



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)$$

Issues with the Vanilla RNNs

- In the same way a product of k real numbers can shrink to zero or explode to infinity, so can a product of matrices

- It is sufficient for $\lambda_1 < 1/\gamma$, where λ_1 is the largest singular value of W, for the **vanishing gradients** problem to occur and it is necessary for **exploding gradients** that $\lambda_1 > 1/\gamma$, where $\gamma = 1$ for the tanh non-linearity and $\gamma = 1/4$ for the sigmoid non-linearity [1]

- Exploding gradients are often controlled with gradient element-wise or norm clipping

[1] On the difficulty of training recurrent neural networks, Pascanu *et al.*, 2013

Long Short-Term Memory (LSTM)[1]

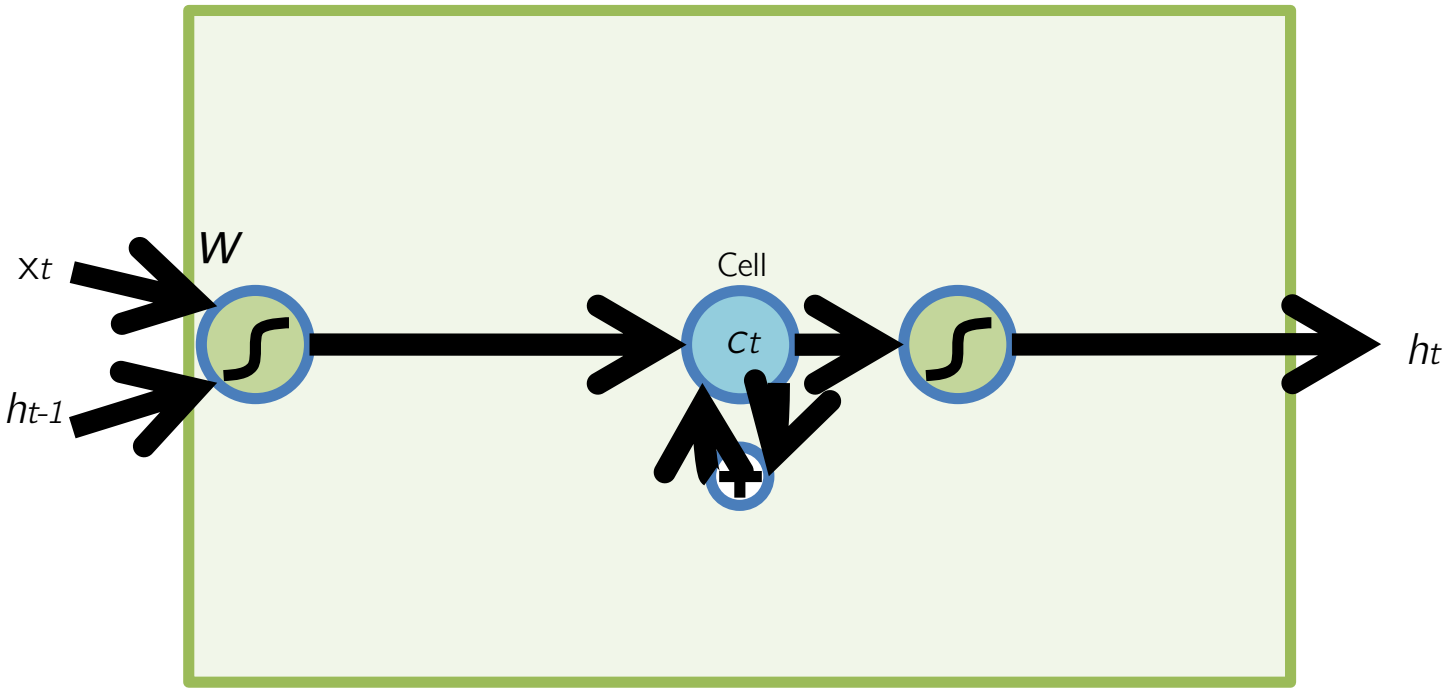
- The LSTM uses the idea of "Constant Error Flow" for RNNs to create a "Constant Error Carousel" (CEC) which ensures that gradients don't decay

- The key component is a memory cell that acts like an accumulator (contains the identity relationship) over time

- Instead of computing new state as a matrix product with the old state, it rather computes the difference between them. Expressivity is the same, but gradients are better behaved

[1] Long Short-Term Memory, Hochreiter *et al.*, 1997

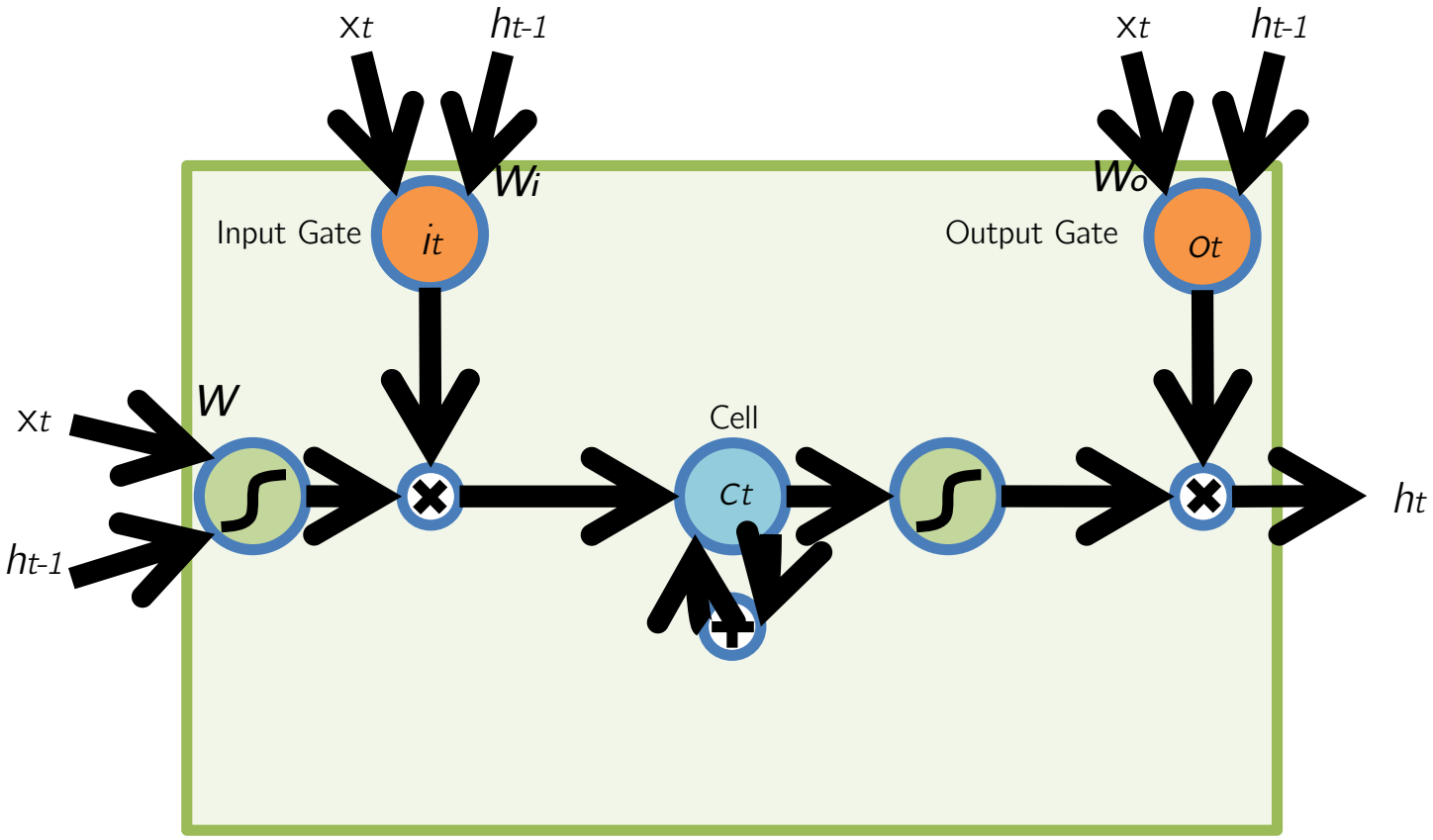
The LSTM Idea



$$c_t = c_{t-1} + \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \qquad h_t = \tanh c_t$$

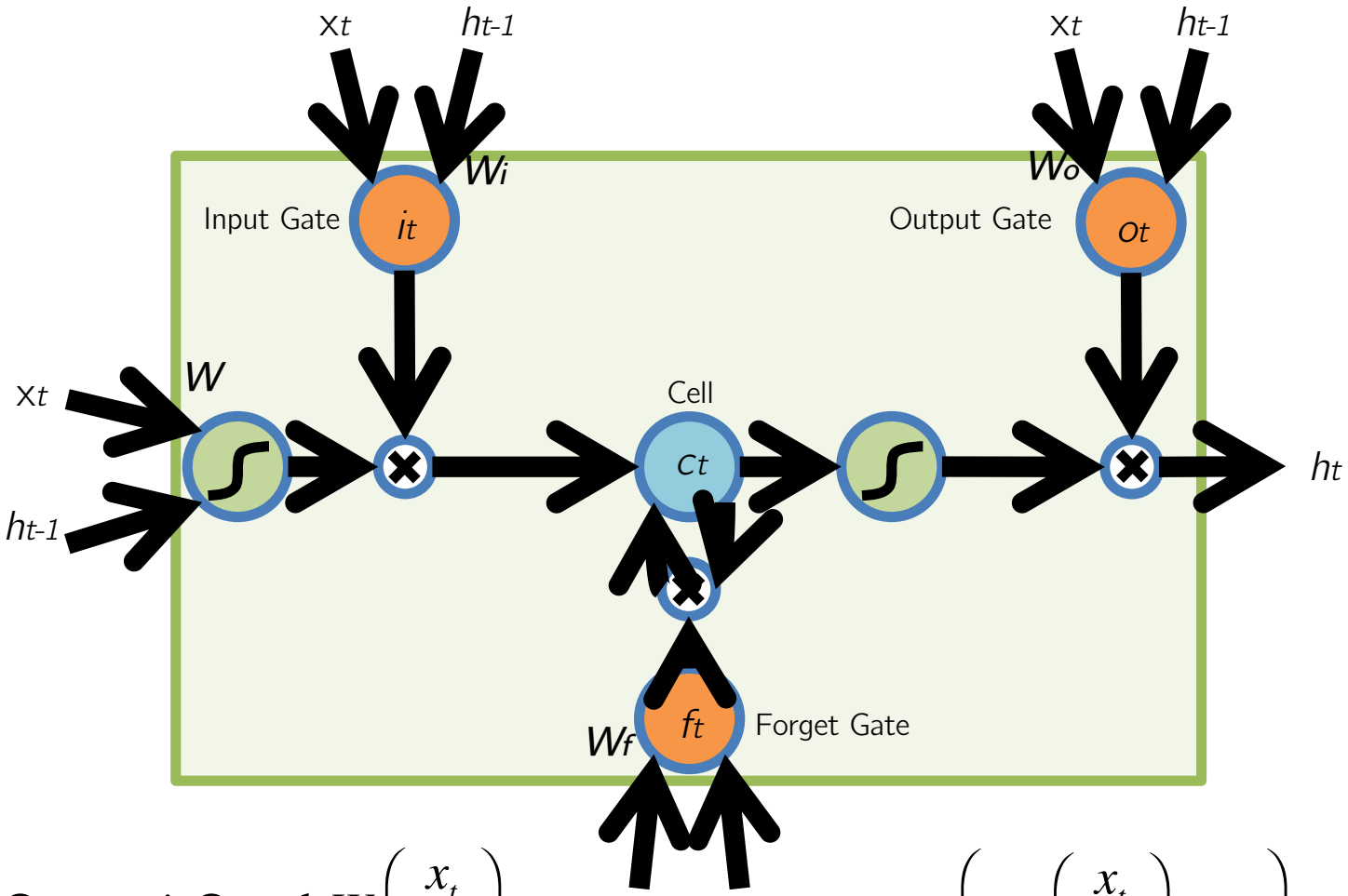
* Dashed line indicates time-lag

The Original LSTM Cell



$$c_t = c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \quad h_t = o_t \otimes \tanh c_t \quad i_t = \sigma\left(W_i \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_i \right)$$ Similarly for o_t

The Popular LSTM Cell



$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

LSTM – Forward/Backward

Go To: Illustrated LSTM Forward and Backward Pass

Summary (so far…)

- RNNs allow for processing of variable length inputs and outputs by maintaining state information across time steps
- Various Input-Output scenarios are possible (Single/Multiple)

- Vanilla RNNs are improved upon by LSTMs which address the vanishing gradient problem through the CEC
- Exploding gradients are handled by gradient clipping

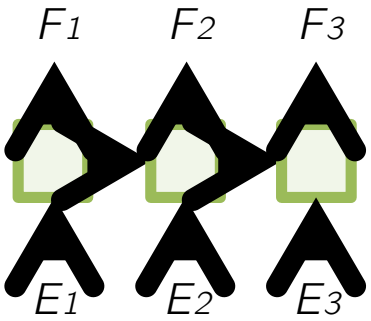
- More complex architectures can be found online and/or the tutorial material for you to play…

Some RNN Variants

Jerry Spanakis

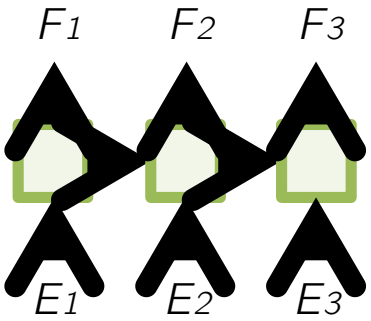
Class Quiz

- Consider the problem of translation of English to Dutch
- E.g. What is your name → Wat is jouw naam
- Is the below architecture suitable for this problem?



Class Quiz

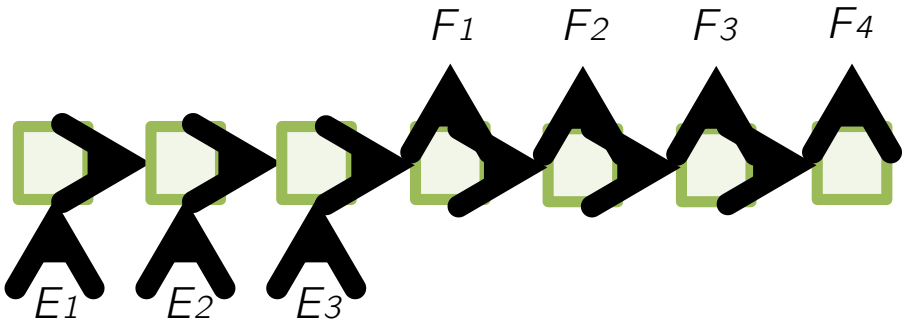
- Consider the problem of translation of English to Dutch
- E.g. What is your name → Wat is jouw naam
- Is the below architecture suitable for this problem?



- No, sentences might be of different length and words might not align. Need to see entire sentence before translating

Class Quiz

- Consider the problem of translation of English to Dutch
- E.g. What is your name → Wat is jouw naam
- Sentences might be of different length and words might not align. Need to see entire sentence before translating

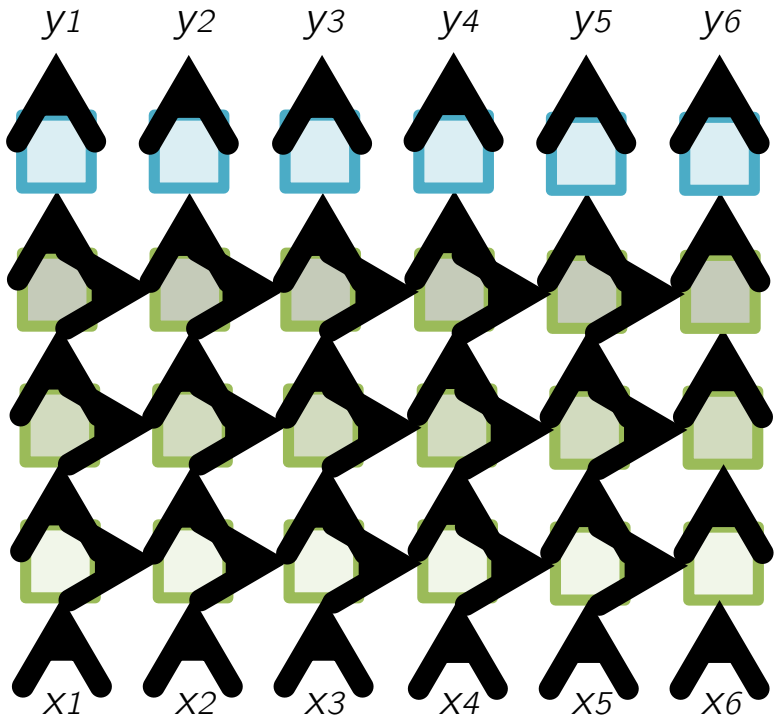


- Input-Output nature depends on the structure of the problem at hand

Seq2Seq Learning with Neural Networks, Sutskever *et al.*, 2014

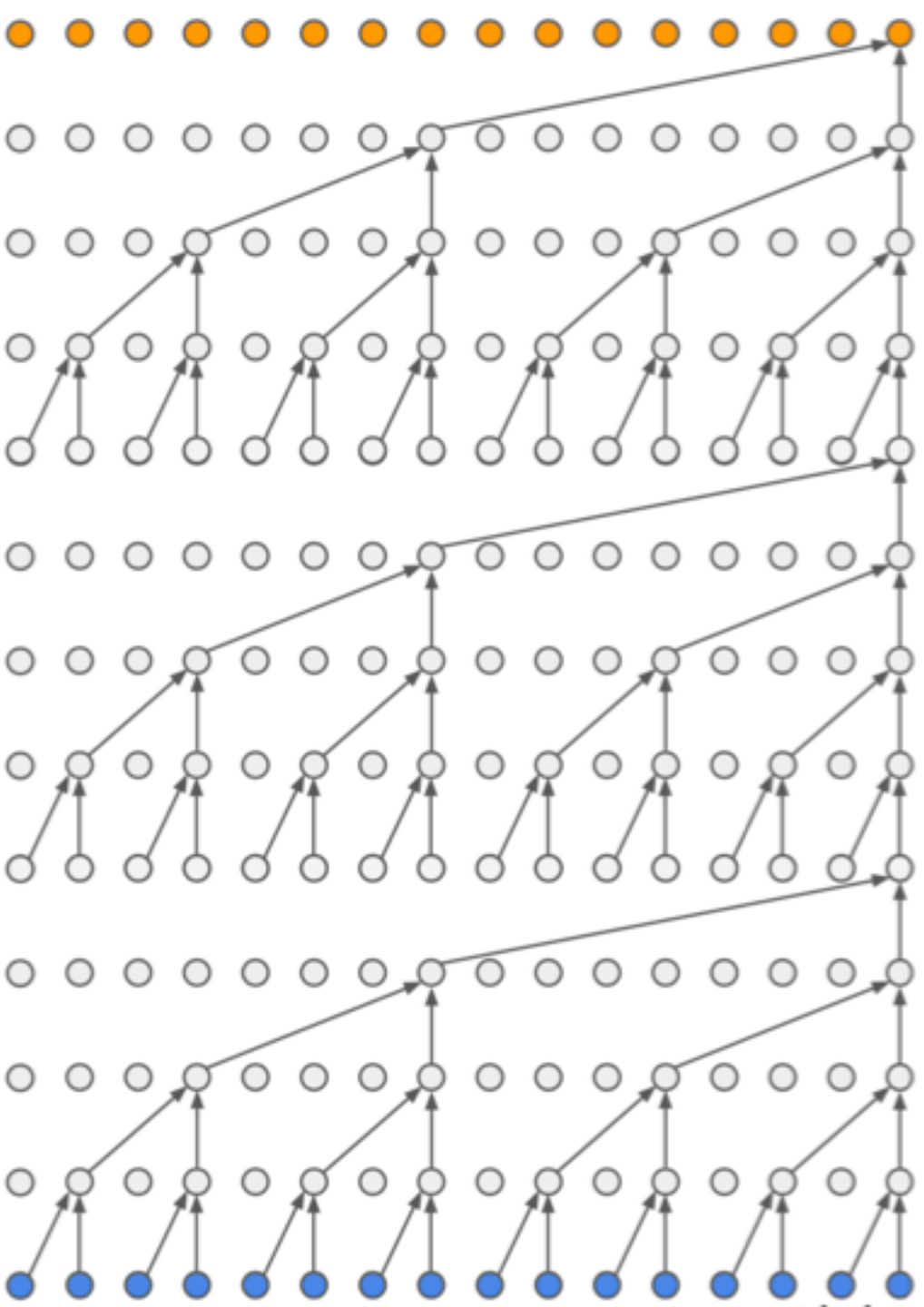
Multi-layer RNNs

- We can also design RNNs with multiple hidden layers



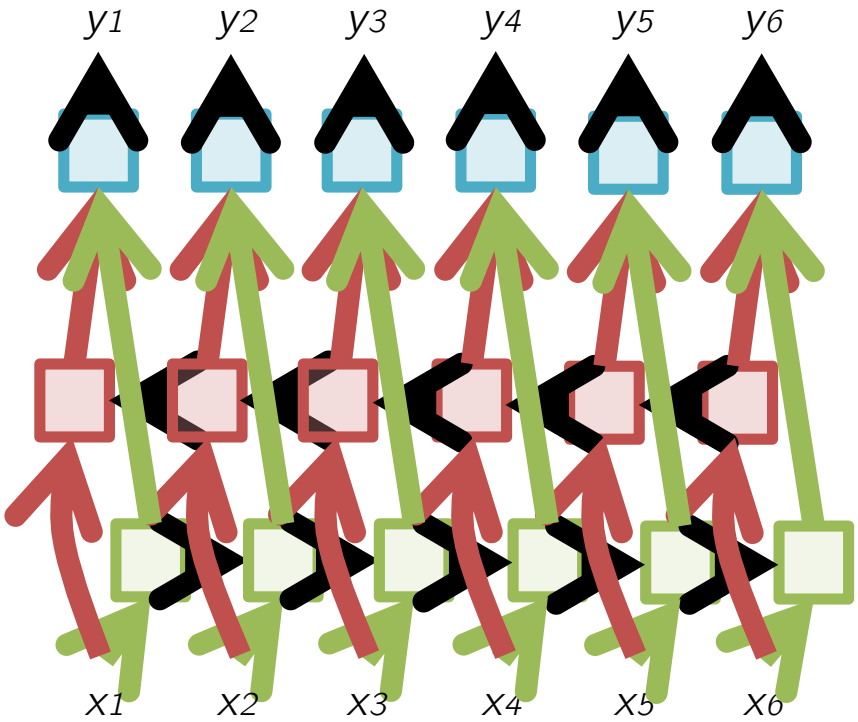
Think big here: Skip connections across layers, across time,…

Multiple Stacks



Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction



- Popular in speech recognition

LSTM: A Search Space Odyssey

- Tested the following variants, using LSTM as standard:
  1. No Input Gate (NIG)
  2. No Forget Gate (NFG)
  3. No Output Gate (NOG)
  4. No Input Activation Function (NIAF)
  5. No Output Activation Function (NOAF)
  6. No Peepholes (NP)
  7. Coupled Input and Forget Gate (CIFG)
  8. Full Gate Recurrence (FGR)
- On the tasks of:
  - Timit Speech Recognition: Audio frame to 1 of 61 phonemes
  - IAM Online Handwriting Recognition: Sketch to characters
  - JSB Chorales: Next-step music frame prediction

LSTM: A Search Space Odyssey, Greff *et al.*, 2015

LSTM: A Search Space Odyssey

- The standard LSTM performed reasonably well on multiple datasets and none of the modifications significantly improved the performance
- Coupling gates and removing peephole connections simplified the LSTM without hurting performance much
- The forget gate and output activation are crucial

- Found interaction between learning rate and network size to be minimal – indicates calibration can be done using a small network first

LSTM: A Search Space Odyssey, Greff *et al.*, 2015

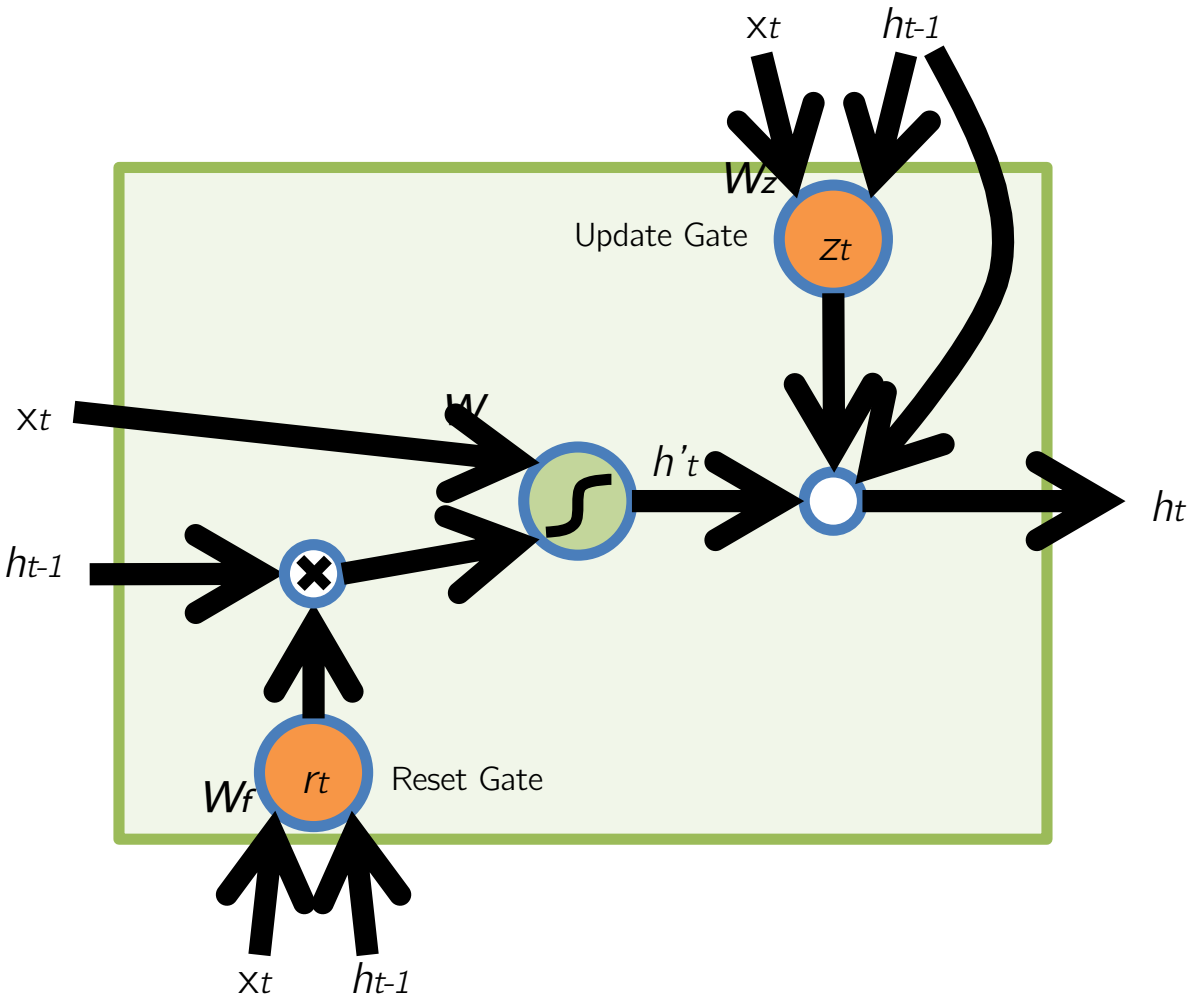
Gated Recurrent Unit (GRU)

- A very simplified version of the LSTM
  - Merges forget and input gate into a single 'update' gate
  - Merges cell and hidden state

- Has fewer parameters than an LSTM and has been shown to outperform LSTM on some tasks

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Cho *et al.*, 2014

GRU



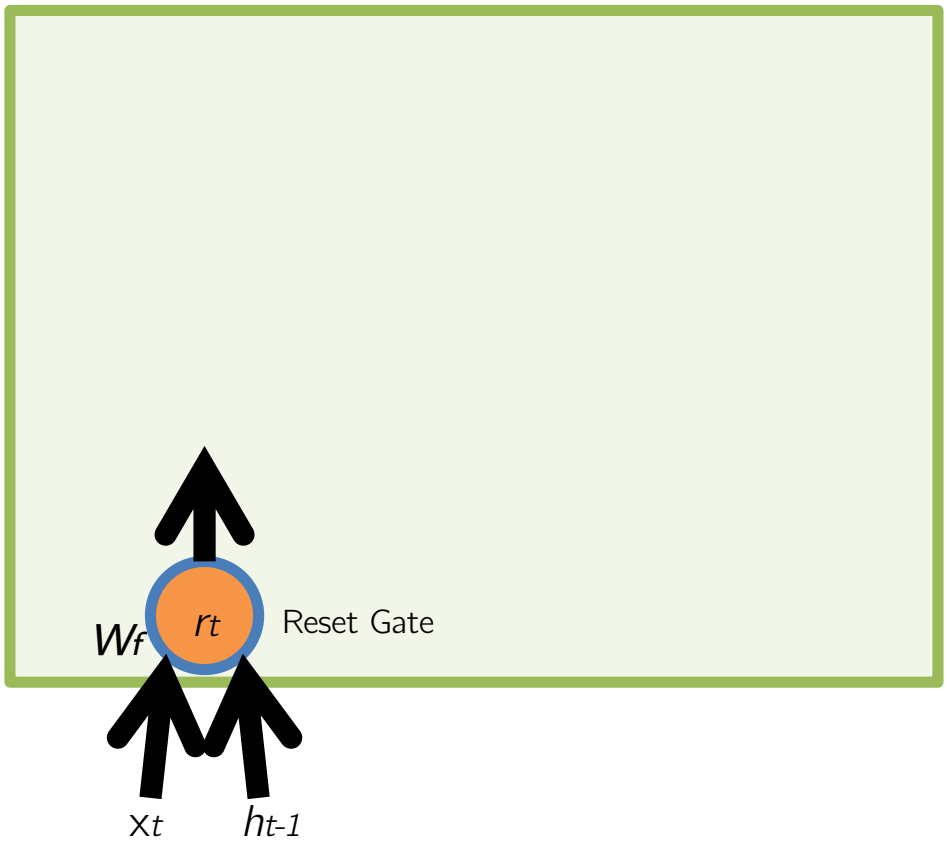
$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes h_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

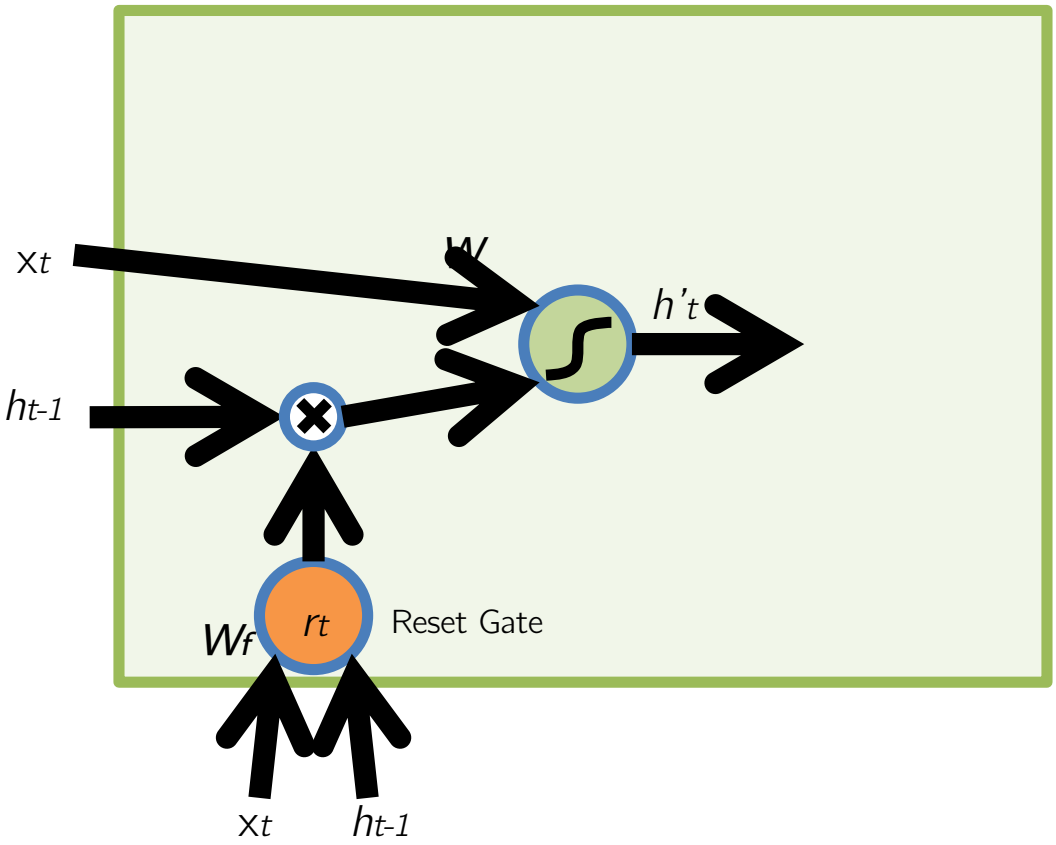
$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes h'_t$$

GRU



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

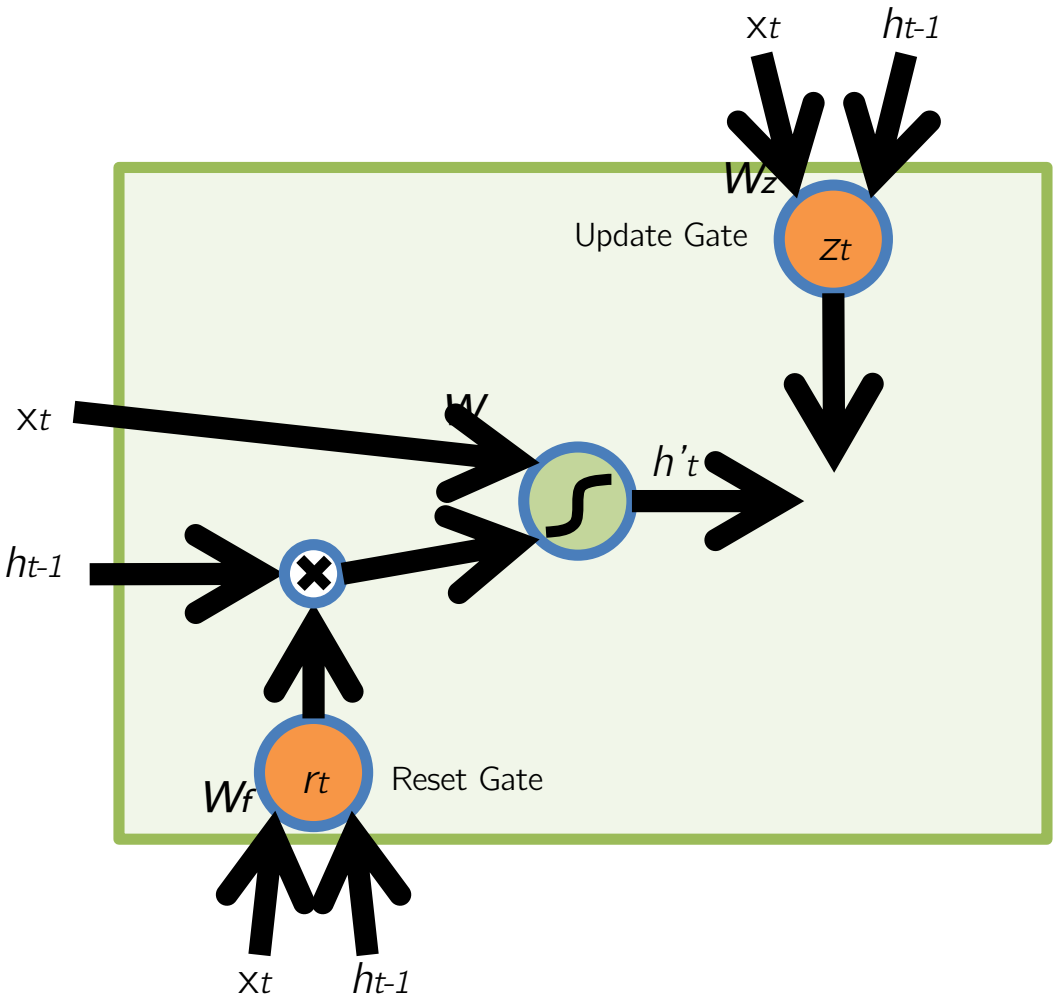
GRU



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes h_{t-1} \end{pmatrix}$$

GRU

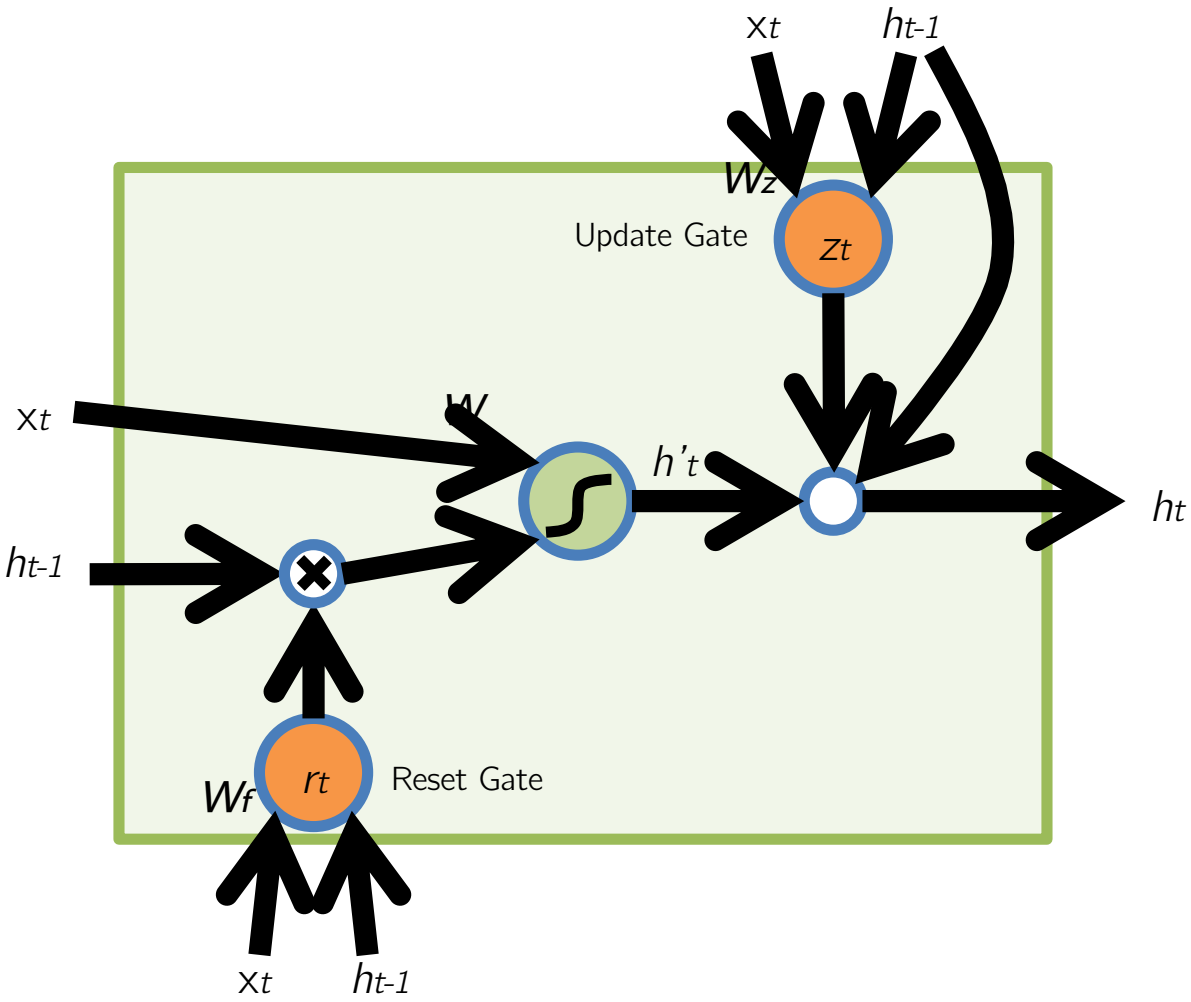


$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes h_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

GRU



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right)$$

$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes h_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right)$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes h'_t$$

An Empirical Exploration of Recurrent Network Architectures

- Given the rather ad-hoc design of the LSTM, the authors try to determine if the architecture of the LSTM is optimal

- They use an evolutionary search for better architectures

An Empirical Exploration of Recurrent Network Architectures, Jozefowicz *et al.*, 2015

Evolutionary Architecture Search

- A list of top-100 architectures so far is maintained, initialized with the LSTM and the GRU
- The GRU is considered as the baseline to beat
- New architectures are proposed, and retained based on performance ratio with GRU

- All architectures are evaluated on 3 problems
  - Arithmetic: Compute digits of sum or difference of two numbers provided as inputs. Inputs have distractors to increase difficulty **3**e**36**d**9-**h**1**h**39**f**94**eeh**43**keg**3**c = 3369 – 13994433 = -13991064
  - XML Modeling: Predict next character in valid XML modeling
  - Penn Tree-Bank Language Modeling: Predict distributions over words

An Empirical Exploration of Recurrent Network Architectures, Jozefowicz *et al.*, 2015

Evolutionary Architecture Search

- At each step
  - Select 1 architecture at random, evaluate on 20 randomly chosen hyperparameter settings.
  - Alternatively, propose a new architecture by mutating an existing one. Choose probability p from [0,1] uniformly and apply a transformation to each node with probability p
    - If node is a non-linearity, replace with {tanh(x), sigmoid(x), ReLU(x), Linear(0, x), Linear(1, x), Linear(0.9, x), Linear(1.1, x)}
    - If node is an elementwise op, replace with {multiplication, addition, subtraction}
    - Insert random activation function between node and one of its parents
    - Replace node with one of its ancestors (remove node)
    - Randomly select a node (node A). Replace the current node with either the sum, product, or difference of a random ancestor of the current node and a random ancestor of A.
  - Add architecture to list based on minimum relative accuracy wrt GRU on 3 different tasks

An Empirical Exploration of Recurrent Network Architectures, Jozefowicz *et al.*, 2015

Evolutionary Architecture Search

- 3 novel architectures are presented in the paper
- Very similar to GRU, but slightly outperform it

- **LSTM initialized with a large positive forget gate bias outperformed both the basic LSTM and the GRU!**

An Empirical Exploration of Recurrent Network Architectures, Jozefowicz *et al.*, 2015

LSTM initialized with large positive forget gate bias?

- Recall

$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

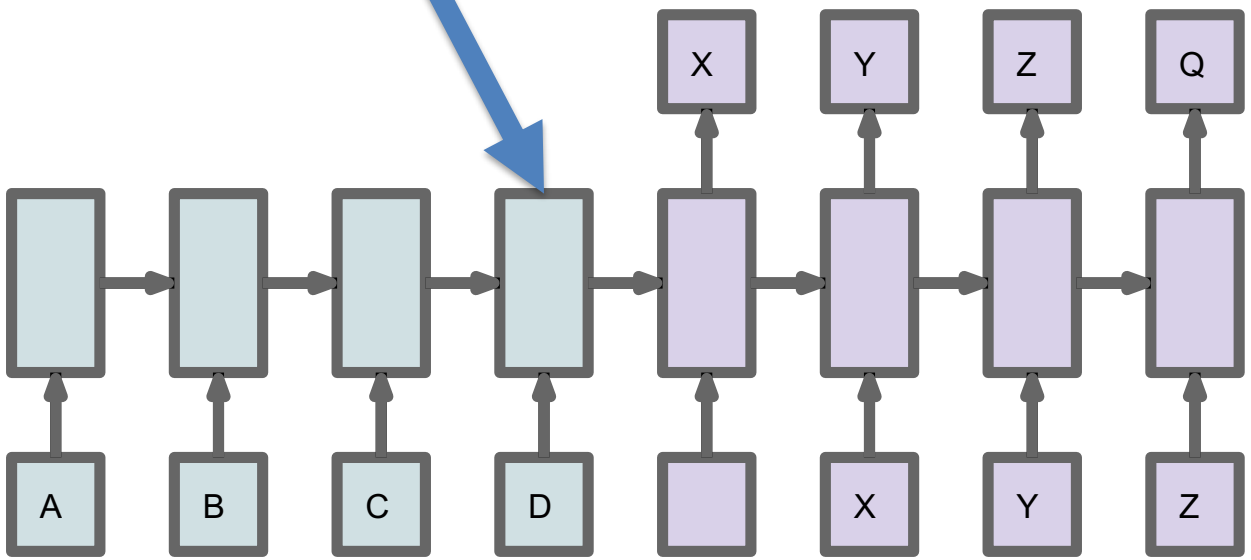
$$\delta c_{t-1} = \delta c_t \otimes f_t$$

- Gradients will vanish if *f* is close to 0. Using a large positive bias ensures that *f* has values close to 1, especially when training begins
- Helps learn long-range dependencies
- Originally stated in Learning to forget: Continual prediction with LSTM, Gers *et al.*, 2000, but forgotten over time

An Empirical Exploration of Recurrent Network Architectures, Jozefowicz *et al.*, 2015

RNN: Issues with long inputs

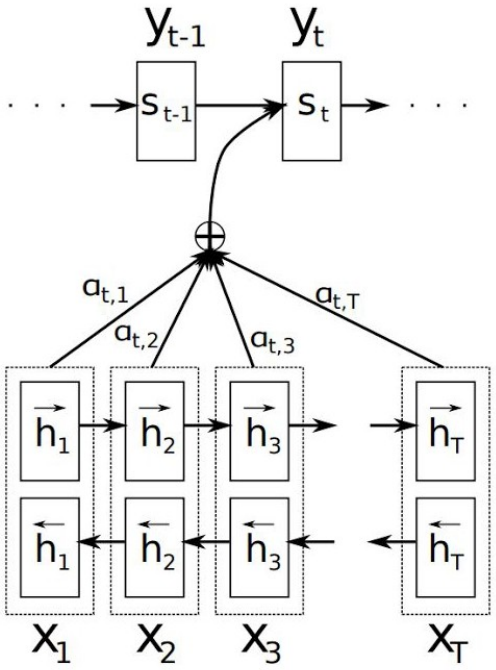
- The same last vector "informs" the entire output
- Needs to capture all the information about the input regardless of length

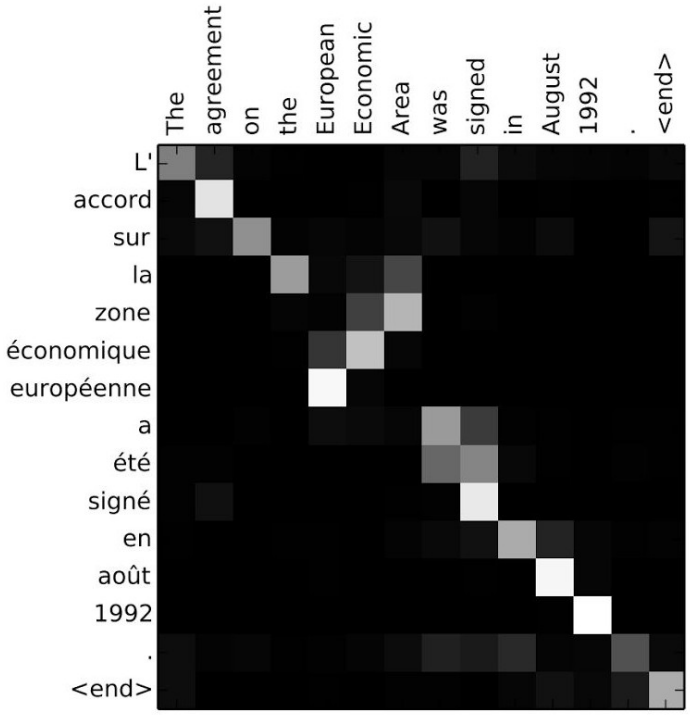


- Can I do better?

Yes, we can!

- Introduce an extra "attention" layer mapping between the input and the ouput

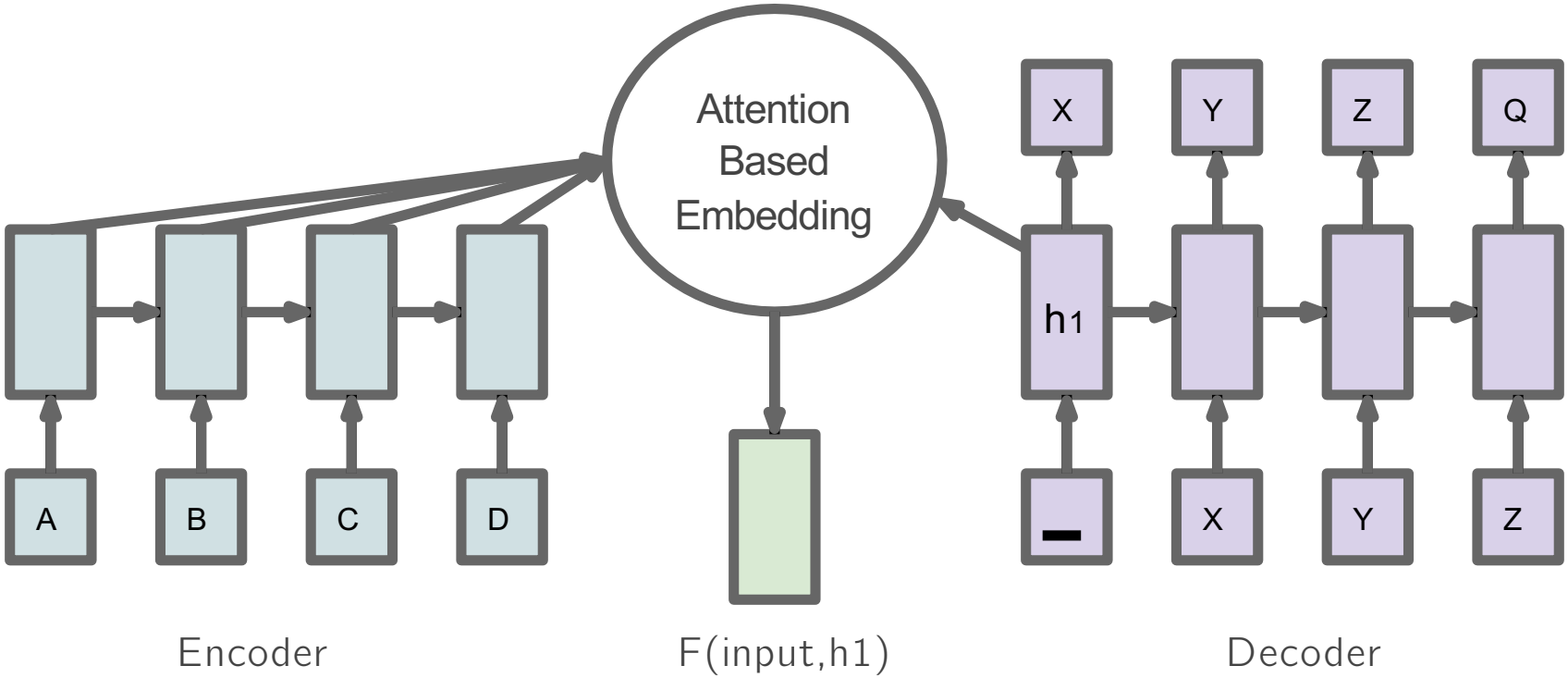




Bahdanau, D., et al. "Neural Machine Translation by Jointly Learning to Align and Translate." *ICLR (2015)*

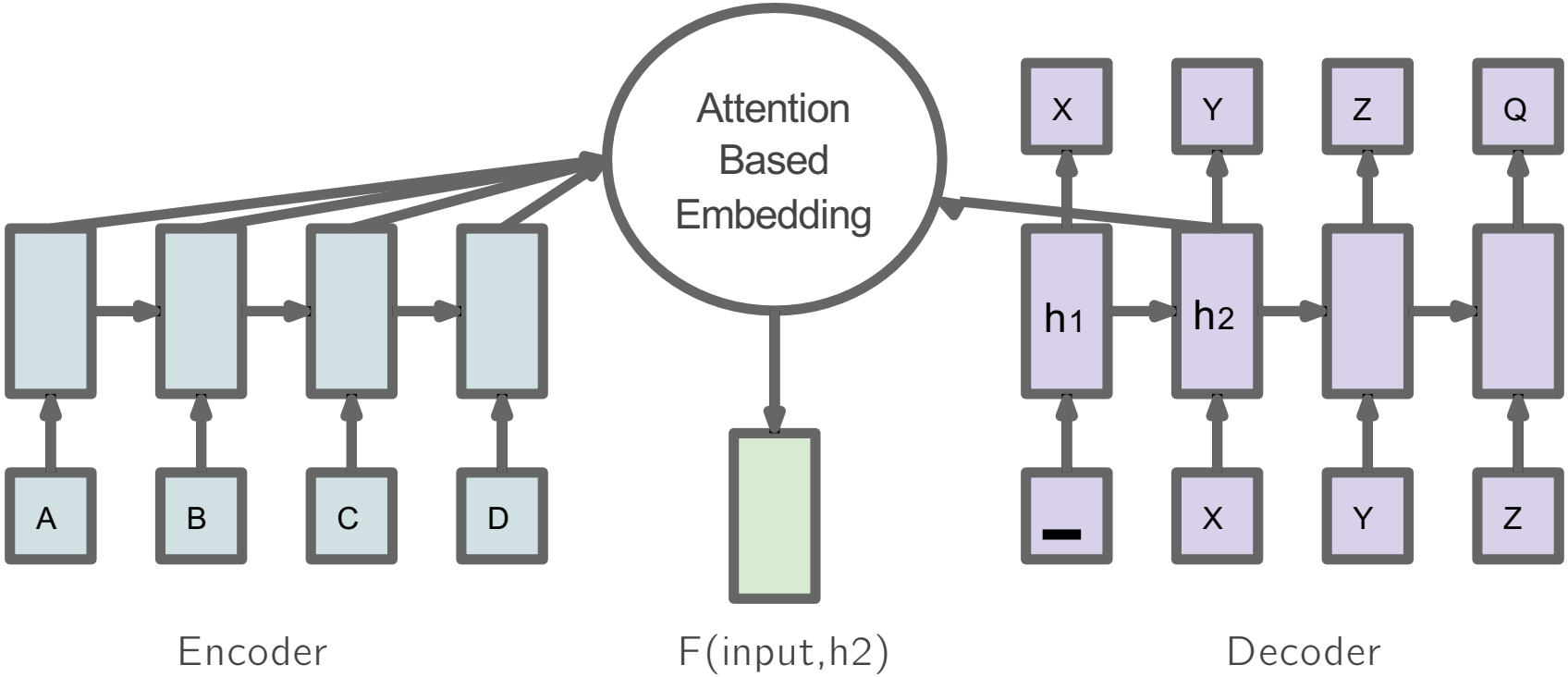
Seq2Seq with Attention

- A different vector computed for every output step



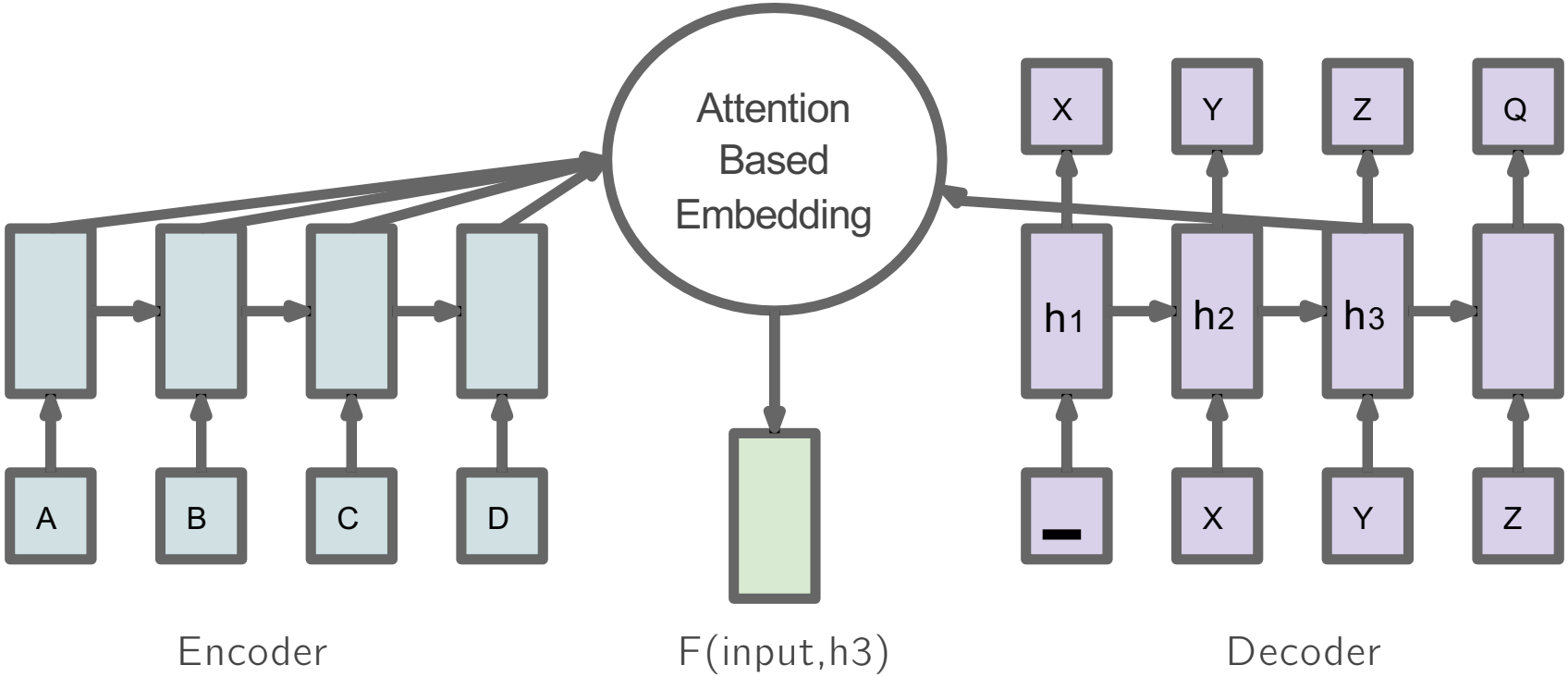
Seq2Seq with Attention

- A different vector computed for every output step



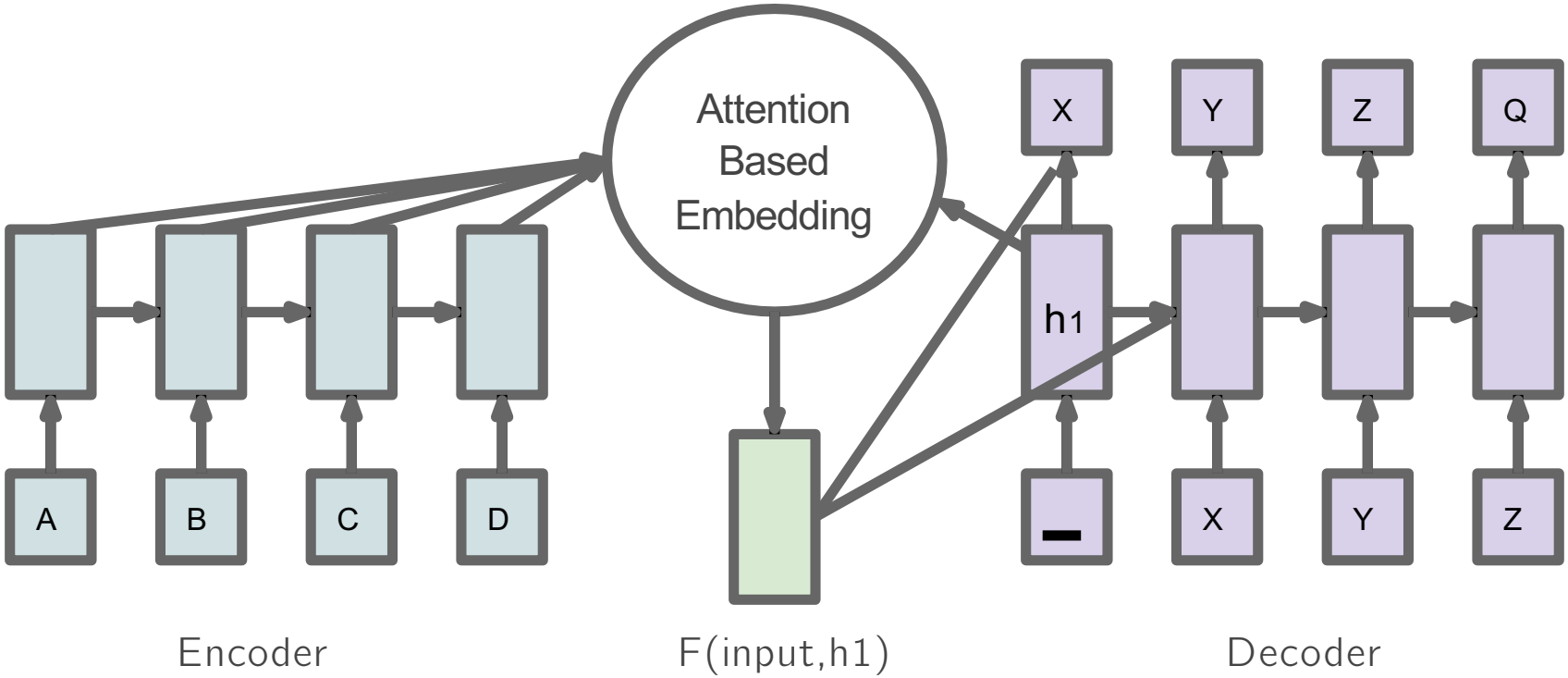
Seq2Seq with Attention

- A different vector computed for every output step



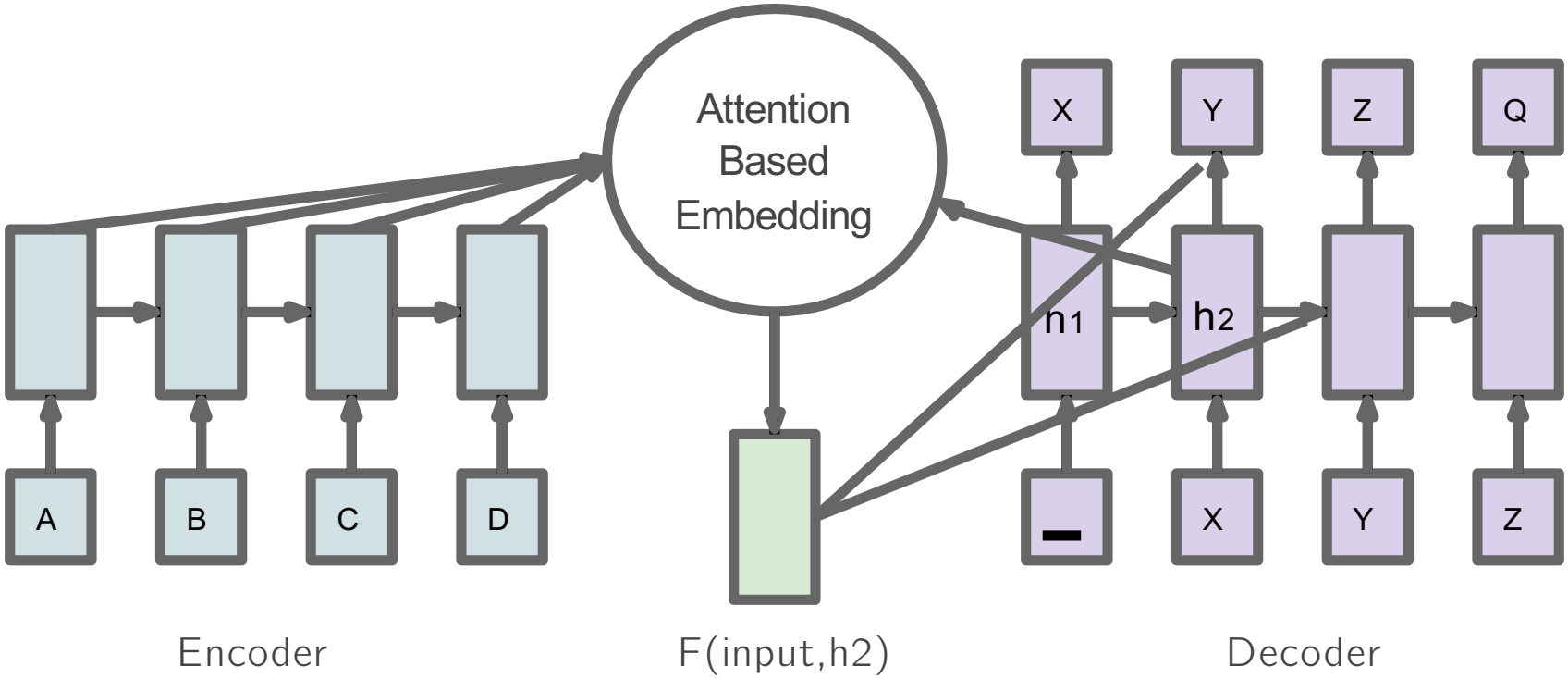
Seq2Seq with Attention

- Attention vector used to predict output and compute next hidden state



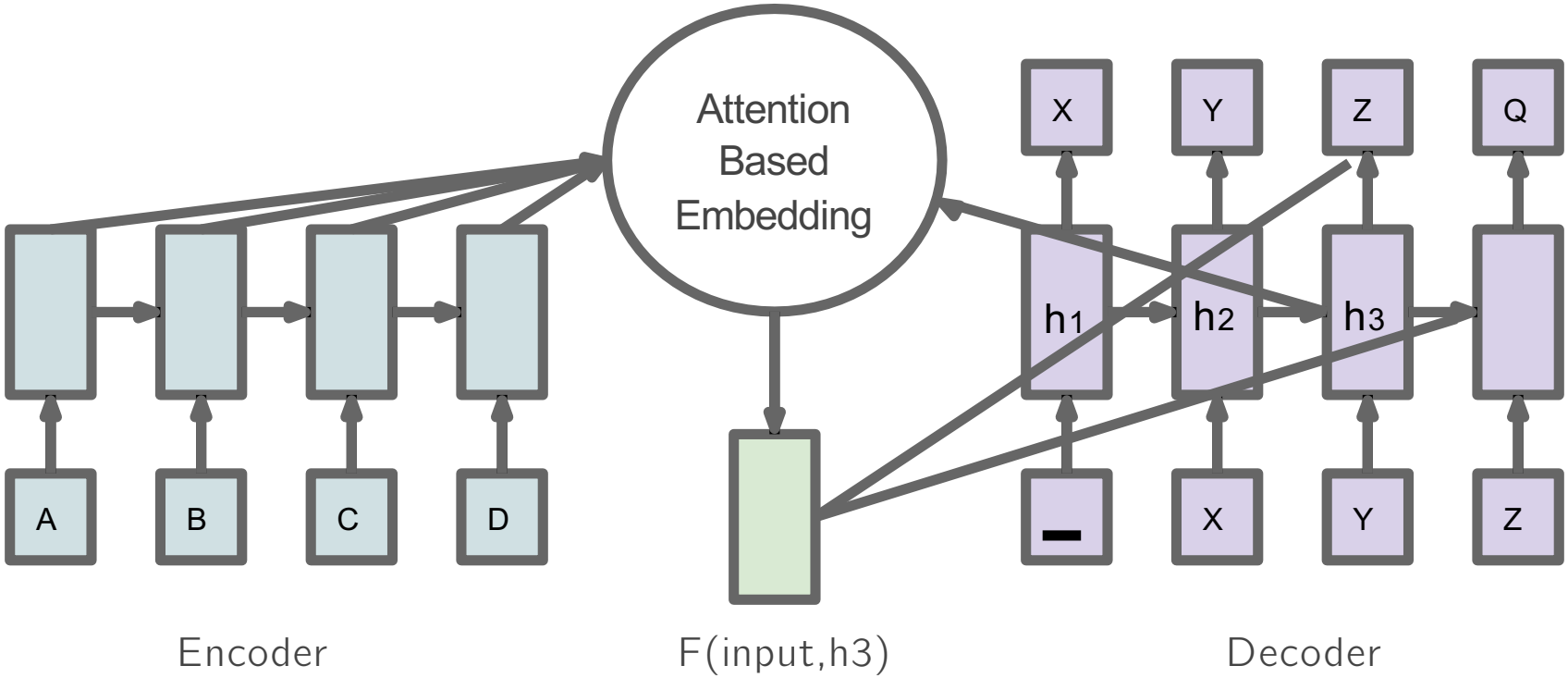
Seq2Seq with Attention

- Attention vector used to predict output and compute next hidden state



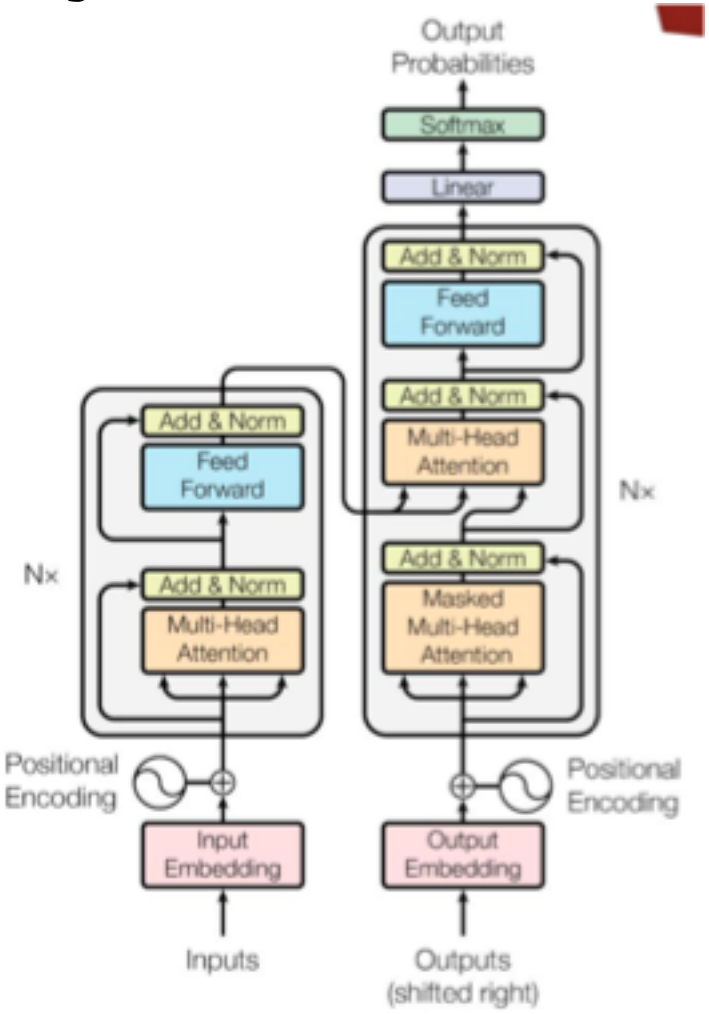
Seq2Seq with Attention

- Attention vector used to predict output and compute next hidden state



Attention is all you need?

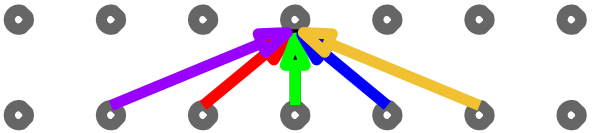
- Not recurrent
- Not convolutional
- Dot-product attention over inputs is masked to preserve causal structure

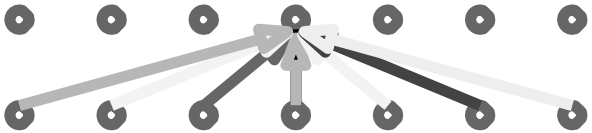


Vaswani, Ashish, et al. "Attention is all you need". In *NIPS*, 2017

Self-Attention

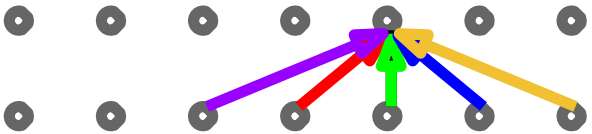
- More powerful than convolutions (which have fixed kernels)
- Less complex than recurrent structures

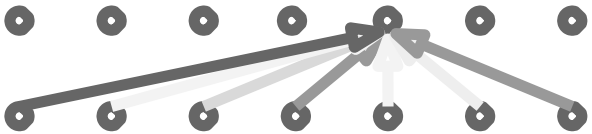




Self-Attention

- More powerful than convolutions (which have fixed kernels)
- Less complex than recurrent structures





Summary & intuitive tricks

- Architectures like the GRU have fewer parameters than the LSTM and **might** perform better
  - An LSTM with large positive forget gate bias works best!
  - Attention models are simpler and they do perform very well!

- Finding the optimal architecture might not the problem you want to solve.
  - Browse the literature and see what works best for the type of problem you want to try.
  - Initialization of parameters is critical but well studied

- If you deal with long sequences
  - Attention or Bigger state or Bi-directional architecture

Introduction to GANs

Jerry Spanakis

Some slide credits (esp. illustrations):
Ian Goofdellow (link)
Oriol Vinyals (link)
Binglin Chen

GANs

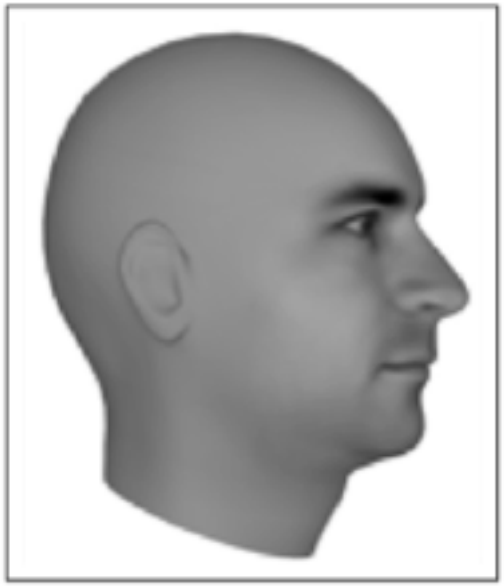
- **Generative**
  - Learn a generative model
- **Adversarial**
  - Trained in an adversarial setting
- **Networks**
  - Use Deep Neural Networks

Why Generative Models?

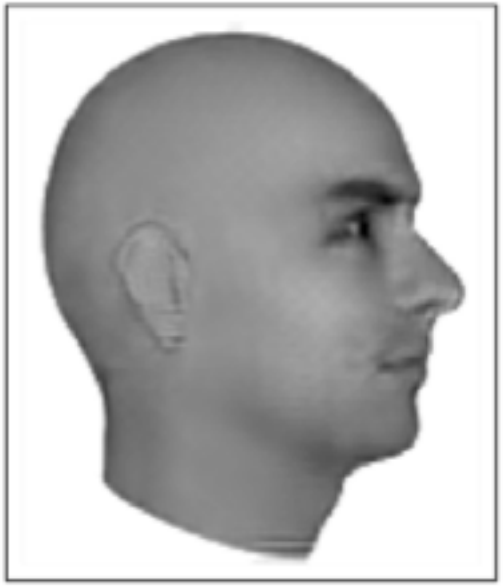
- **CNN, RNN are discriminative models**
  - E.g. given an image (or sequence) **X**, predict a label **Y**
  - Estimates **P(Y|X)**
- **Discriminative models have several key limitations**
  - Can't model **P(X)**, i.e. the probability of seeing a certain image
  - Thus, can't sample from **P(X)**, i.e. **can't generate new images**
- **Generative models (in general) cope with all of above**
  - Can model **P(X)**
  - Can generate new images

Magic of GANs...

Ground Truth



Adversarial



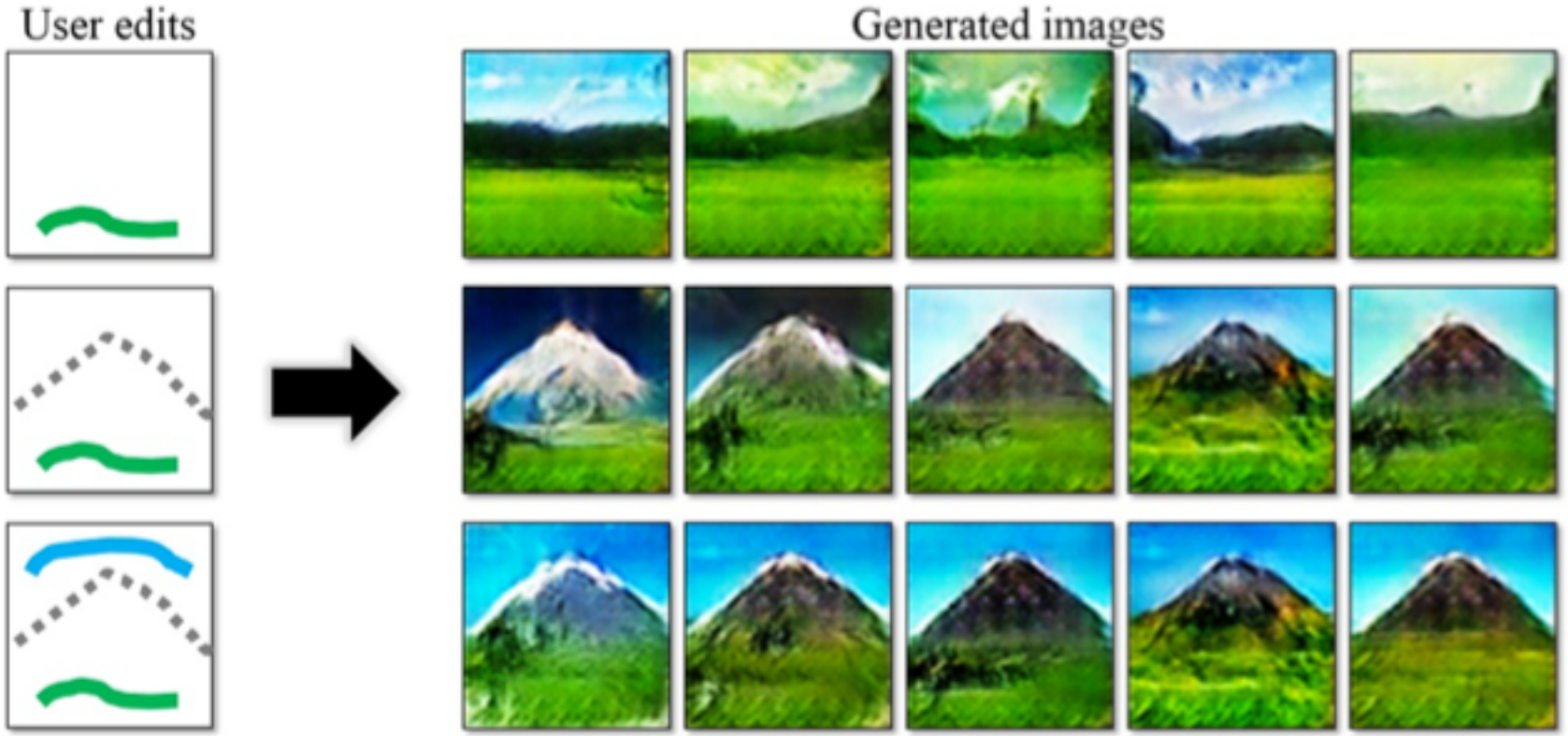
Magic of GANs...

[Ledig. et. al, 2017 “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”]

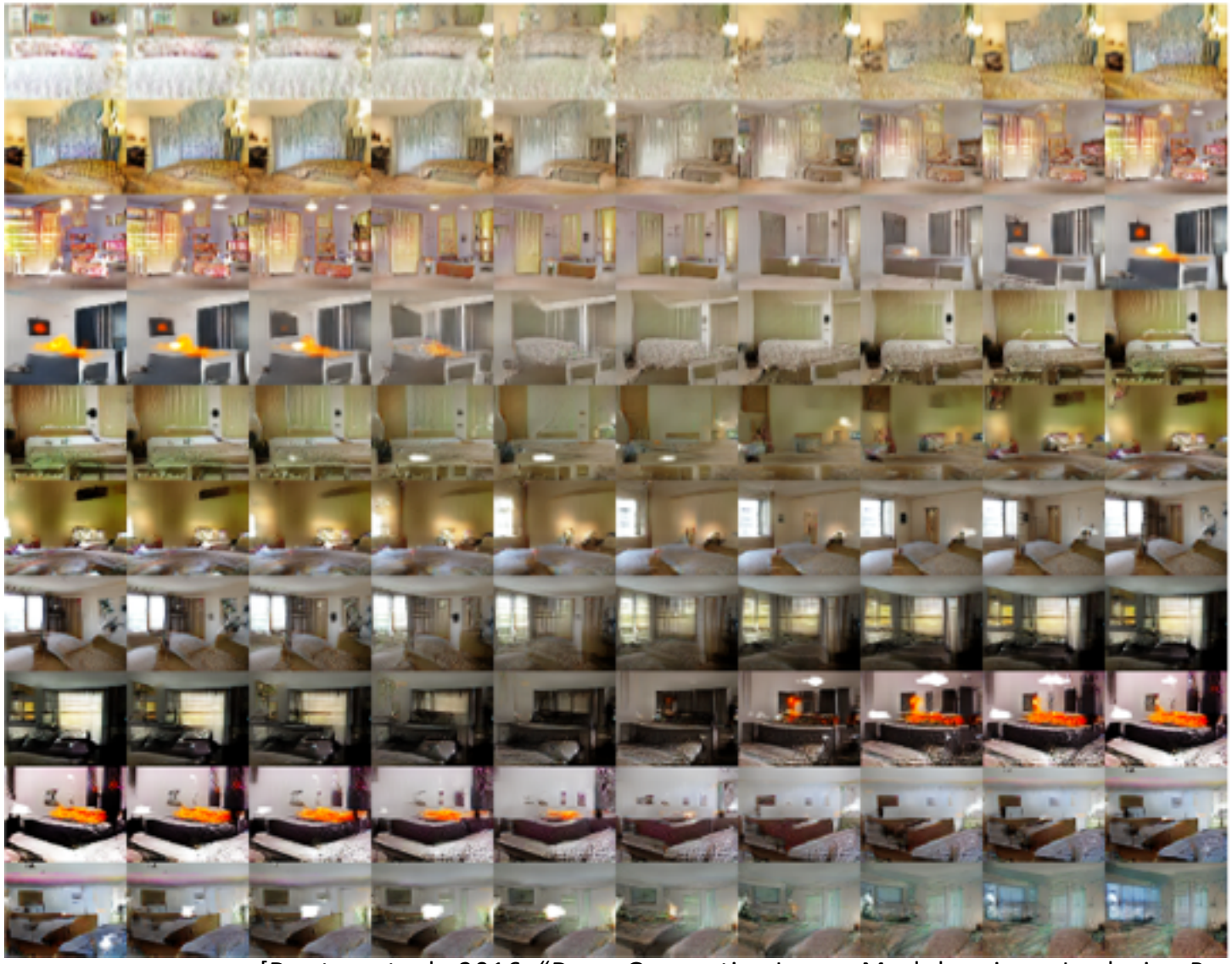
Magic of GANs...

[Work by vue.ai : Online fashion tech startup]

Magic of GANs...



Magic of GANs...



[Denton et. al., 2016, "Deep Generative Image Models using a Laplacian Pyramid of GANs"]

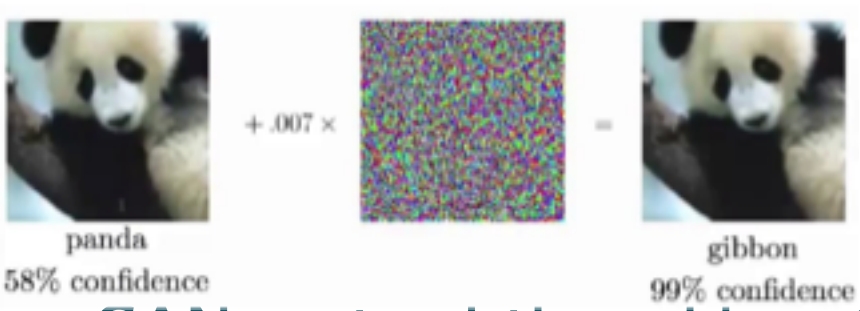
[Radford et. al, 2016, "Unsupervised Representation Learning with DCGANs"]

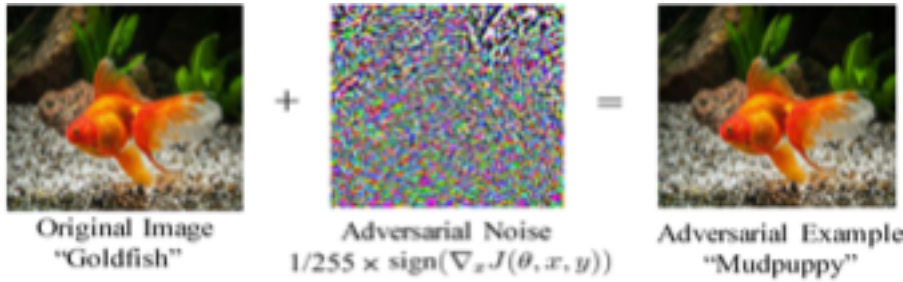
Magic of GANs...

[Karras, et. al. 2017, “Progressive Growing of GANs for Improved Quality, Stability, and Variation”]

Adversarial Training

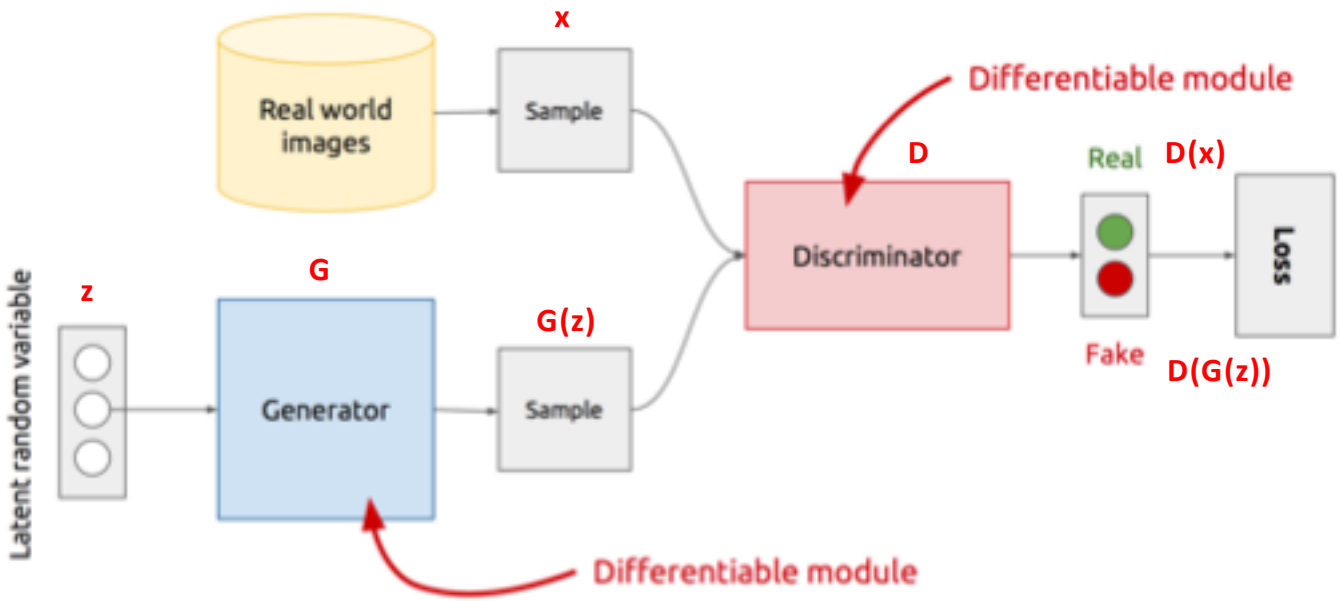
- Basic ideas:
  - We can generate adversarial samples to fool a discriminative model
  - We can use those adversarial samples to make models robust
  - We then require more effort to generate adversarial samples
  - Repeat this and we get better discriminative model





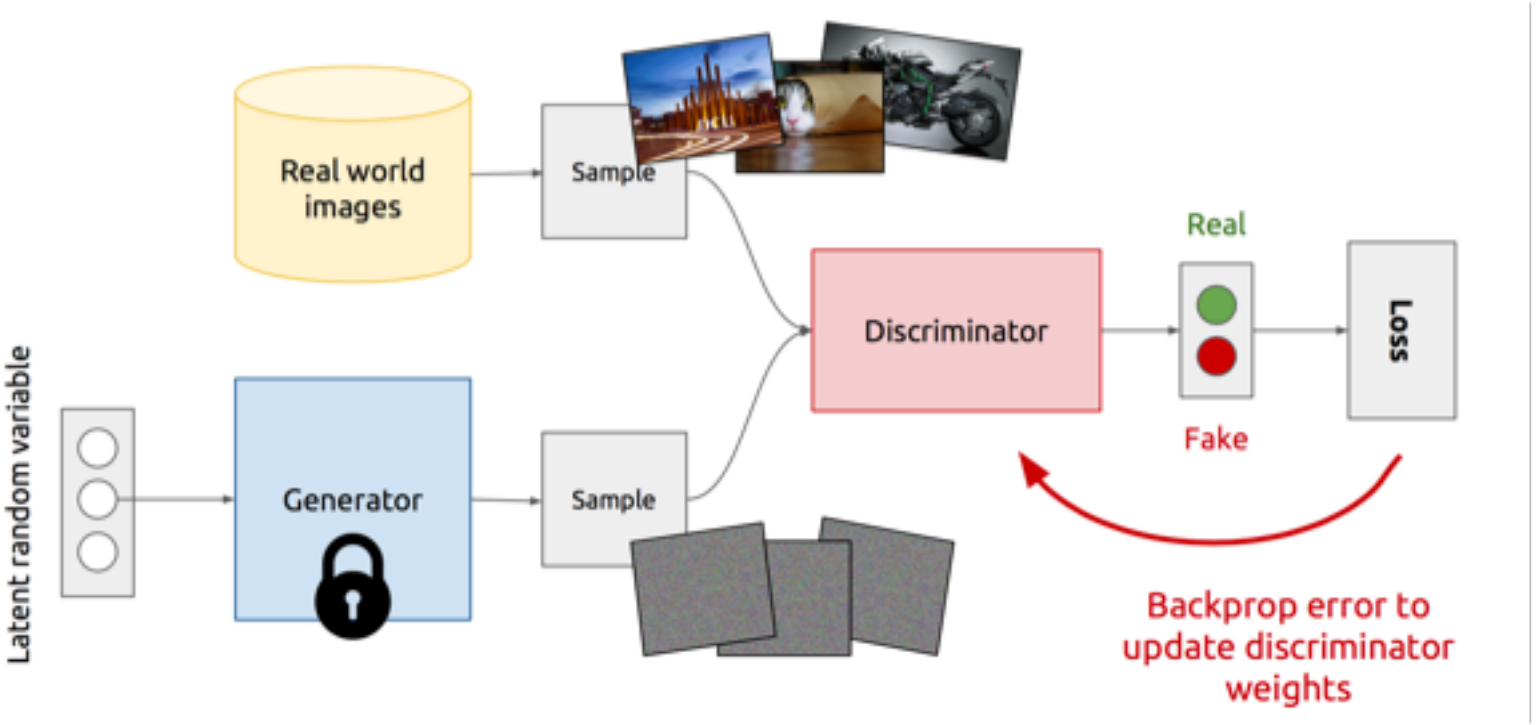
- GANs extend these ideas to generative models:
  - **Generator**: generate fake samples, tries to fool the Discriminator
  - **Discriminator**: tries to distinguish between real and fake samples
  - Train them against each other
  - Repeat this and we get better Generator and Discriminator

GAN's Architecture

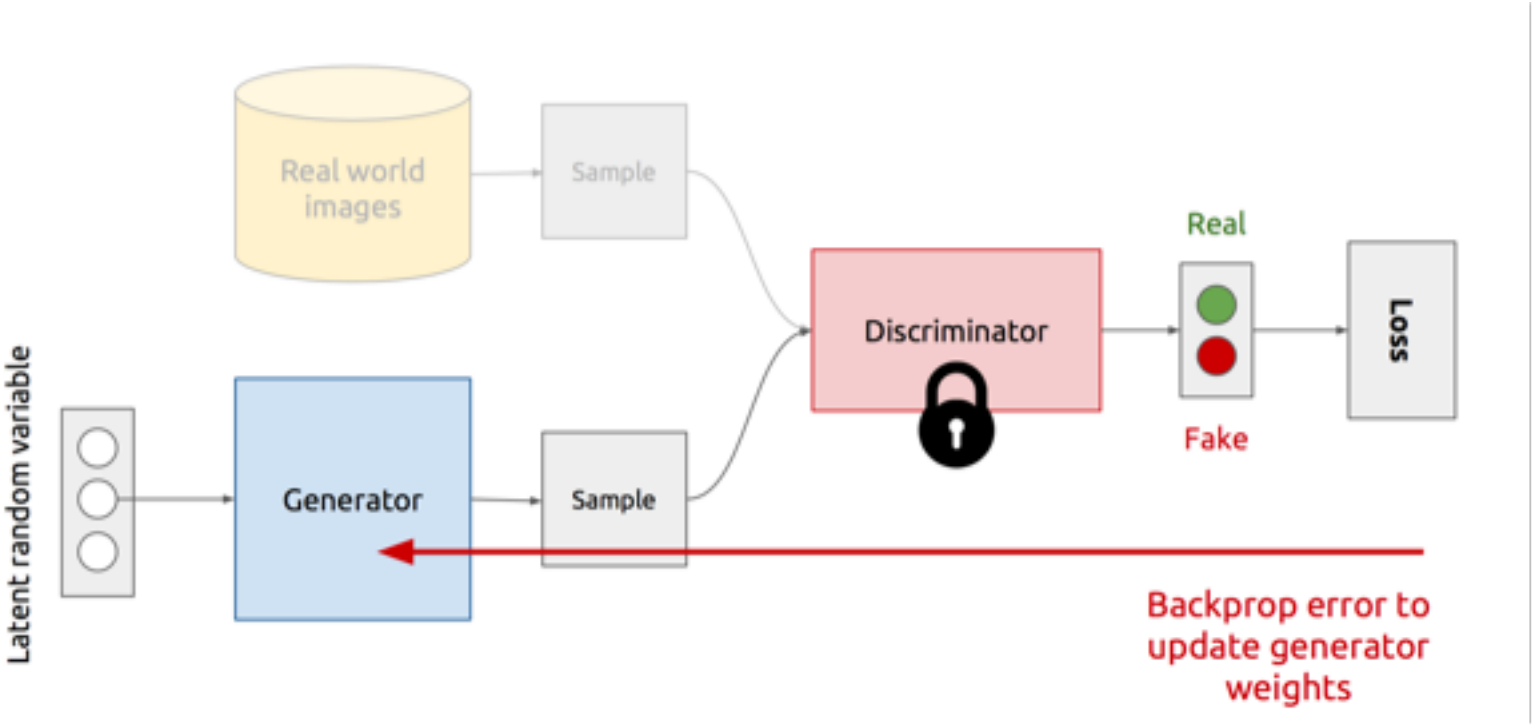


- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

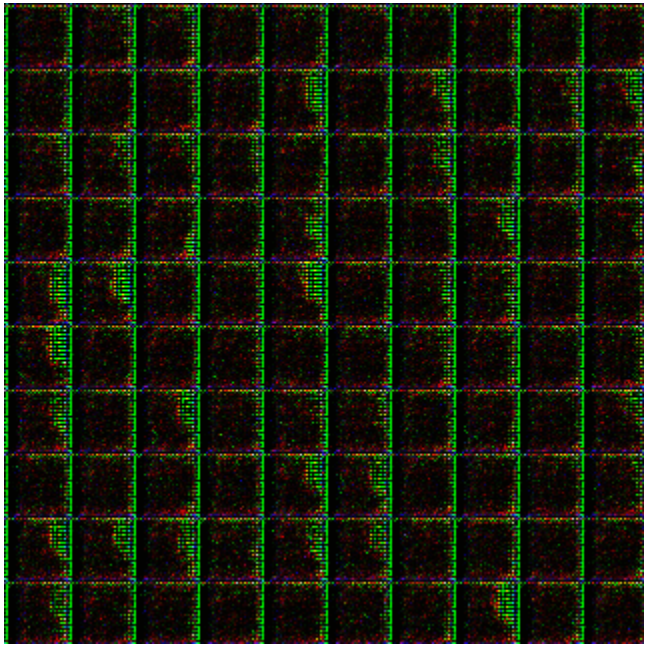
Training Discriminator

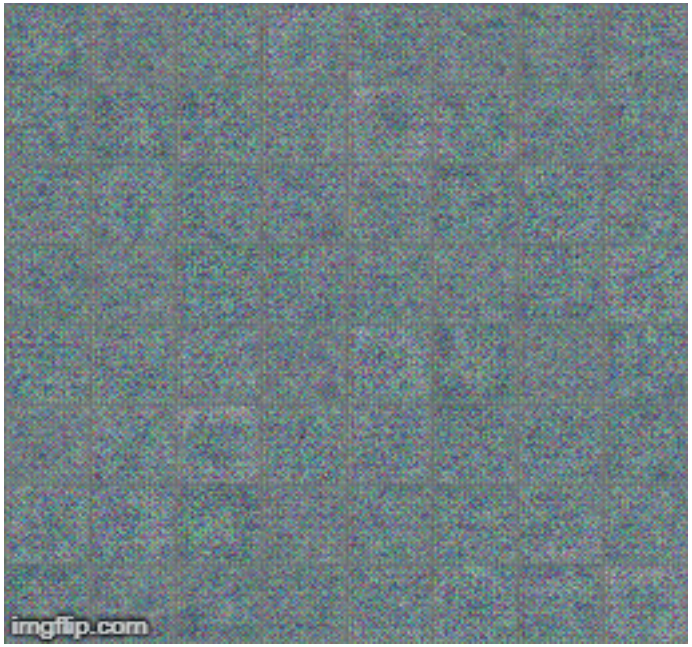


Training Generator



Generator in action





GAN's formulation

$$\min_G \max_D V(D,G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \;\; \forall x$

Vanishing gradient strikes back again...

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

- $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- Gradient goes to 0 if D is confident, i.e. $D\left(G(z)\right) \to 0$

- Minimize $-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]$ for **Generator** instead (keep Discriminator as it is)

Faces

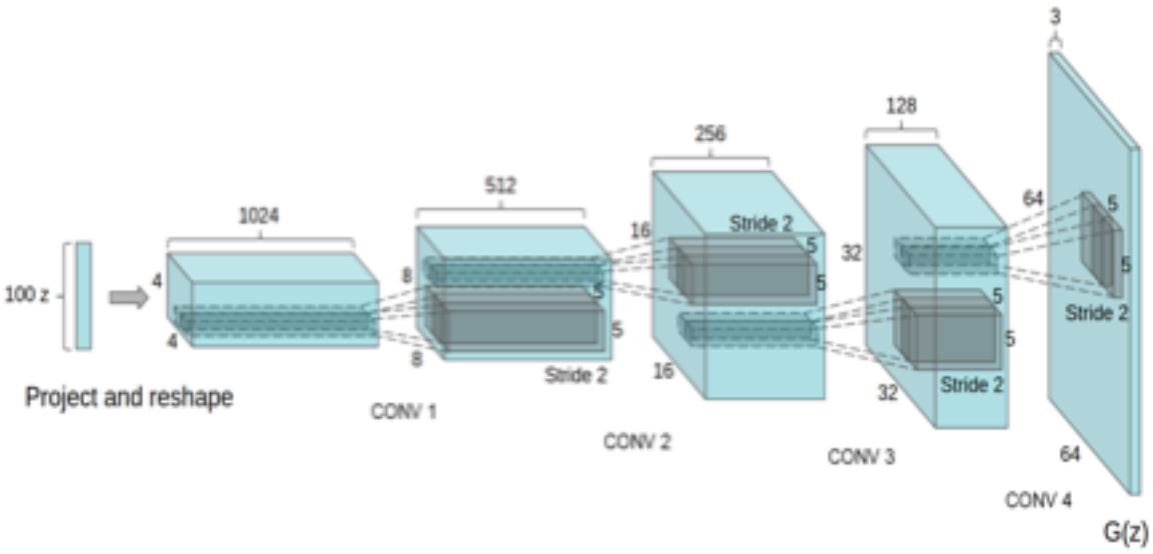
CIFAR

https://www.cs.toronto.edu/~kriz/cifar.html

DCGAN: Bedroom images

Deep Convolutional GANs (DCGANs)

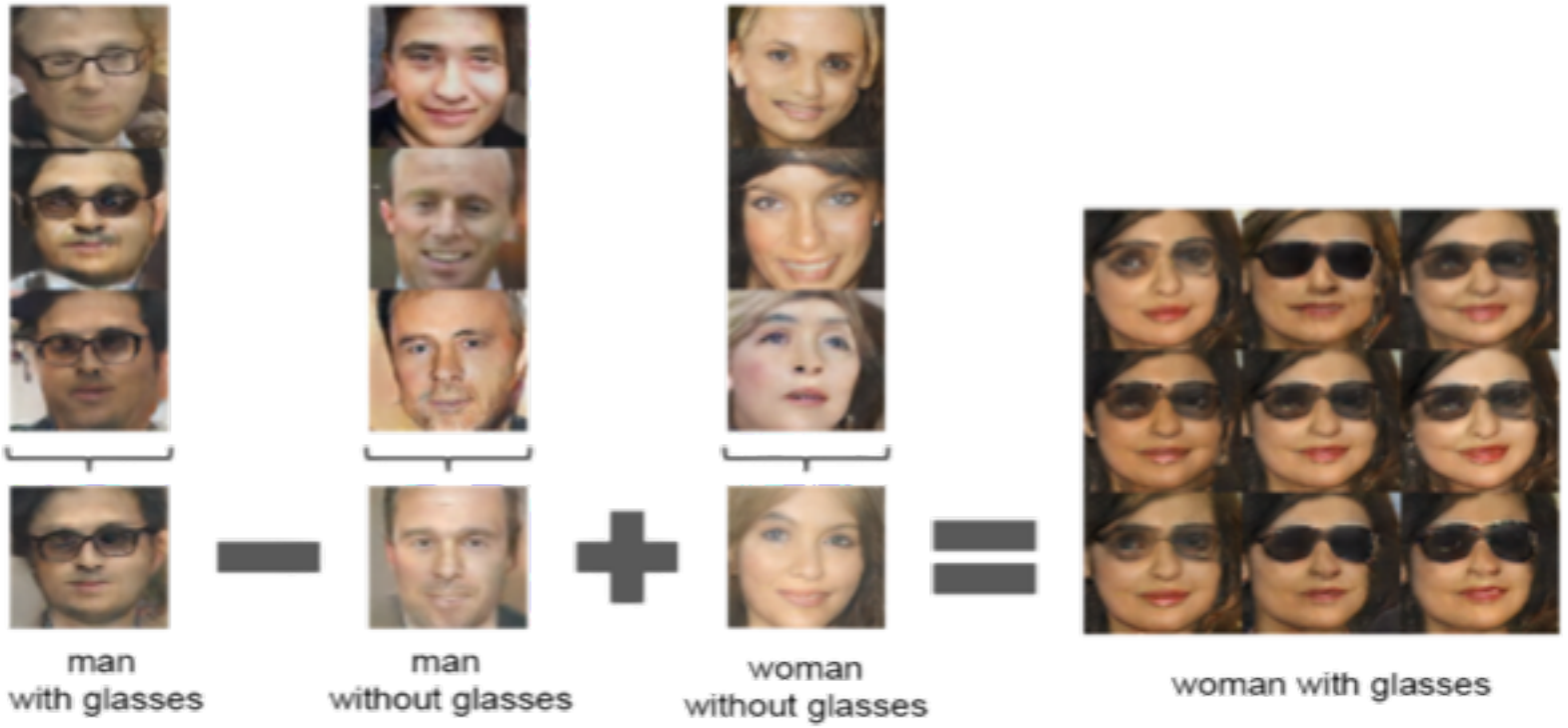
Generator Architecture



Key ideas:

- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

Latent vectors capture interesting patterns...



GANs: Part 2

- Advantages of GANs
- Training Challenges
  - Non-Convergence
  - Mode-Collapse
- Proposed Solutions
  - Supervision with Labels
  - Mini-Batch GANs
- Current trends
  - Wasserstein GANs
  - Conditional GANs
- (fancy) applications

Advantages of GANs

- **Plenty of existing work on Deep Generative Models**
  - Boltzmann Machine
  - Deep Belief Nets
  - Variational AutoEncoders (VAE)

- **Why GANs?**
  - Sampling (or generation) is straightforward.
  - Training doesn't involve Maximum Likelihood estimation.
  - Robust to Overfitting since Generator never sees the training data.
  - Empirically, GANs are good at capturing the modes of the distribution.

Problems with GANs

- **Probability Distribution is Implicit**
  - Not straightforward to compute P(X).
  - Thus **Vanilla GANs** are only good for Sampling/Generation.

- **Training is Hard**
  - Non-Convergence
  - Mode-Collapse

Training Problems

- Non-Convergence
- Mode-Collapse

Non-Convergence

$$\min_{x} \max_{y} V(x, y)$$

Let $V(x, y) = xy$

- State 1: | x > 0 | y > 0 | V > 0 | → | Increase y | Decrease x |
- State 2: | x < 0 | y > 0 | V < 0 | → | Decrease y | Decrease x |
- State 3: | x < 0 | y < 0 | V > 0 | → | Decrease y | Increase x |
- State 4 : | x > 0 | y < 0 | V < 0 | → | Increase y | Increase x |
- State 5: | x > 0 | y > 0 | V > 0 | == State 1 → | Increase y | Decrease x |

- **Deep Learning models (in general) involve a single player**
  - The player tries to maximize its reward (minimize its loss).
  - Use SGD (with Backpropagation) to find the optimal parameters.
  - SGD has convergence guarantees (under certain conditions).
  - **Problem:** With non-convexity, we might converge to local optima.

$$\min_G L(G)$$

- **GANs instead involve two (or more) players**
  - Discriminator is trying to maximize its reward.
  - Generator is trying to minimize Discriminator's reward.

$$\min_G \max_D V(D, G)$$

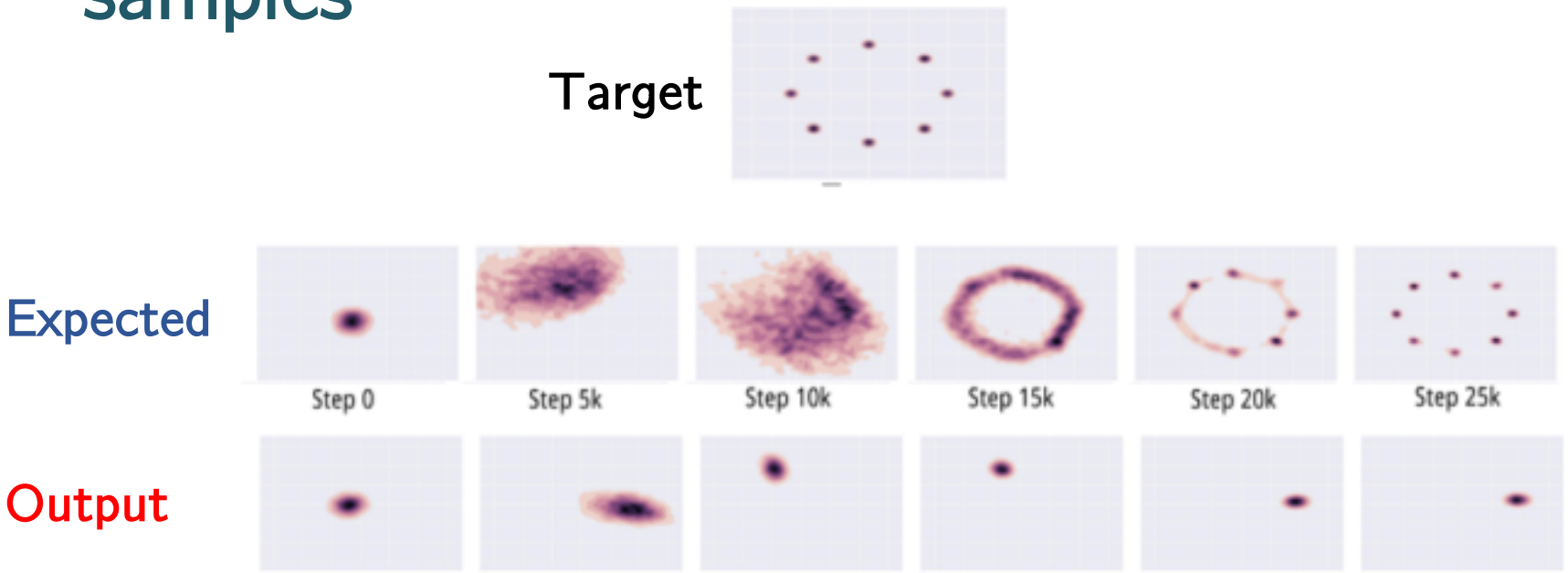
  - SGD was not designed to find the Nash equilibrium of a game.
  - **Problem:** We might not converge to the Nash equilibrium at all.

Problems with GANs

- Non-Convergence
- **Mode-Collapse**

Mode-Collapse

- Generator fails to output diverse samples



Some real examples

Some Solutions

- Mini-Batch GANs
- Supervision with labels

- Recent best-performing GAN:
  Improved Wasserstein-GAN

  https://arxiv.org/abs/1704.00028

Basic (Heuristic) Solutions

- Mini-Batch GANs
- Supervision with labels

How to reward **sample diversity**?

- **At Mode Collapse,**
  - Generator produces good samples, but a very few of them.
  - Thus, Discriminator can't tag them as fake.
- **To address this problem,**
  - Let the Discriminator know about this edge-case.
- **More formally,**
  - Let the Discriminator look at the entire batch instead of single examples
  - If there is lack of diversity, it will mark the examples as fake
- **Thus,**
  - Generator will be forced to produce diverse samples.

Mini-Batch GANs

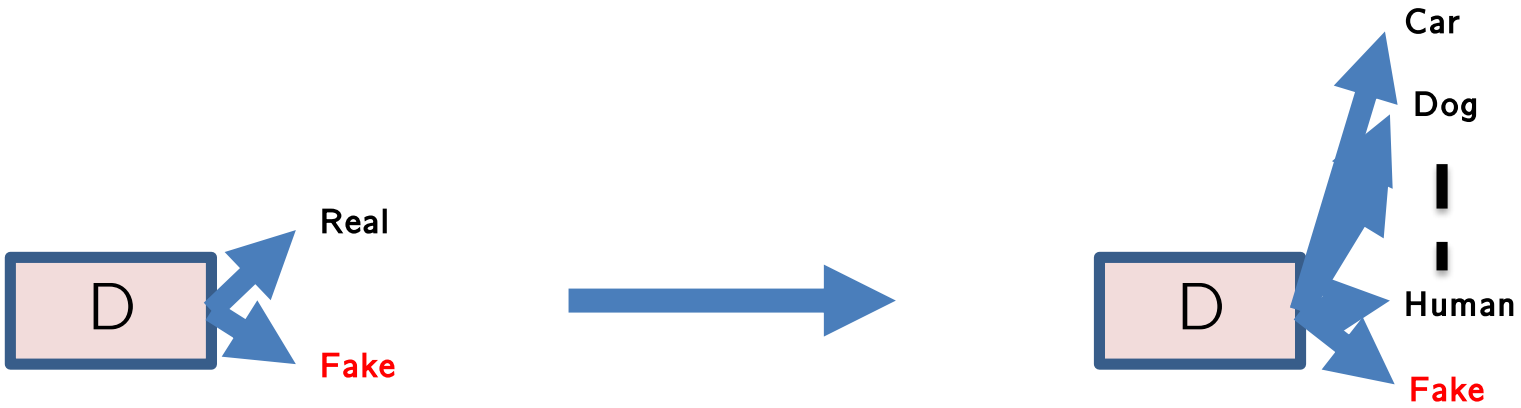
- **Extract features that capture diversity in the mini-batch**
  - For e.g. L2 norm of the difference between all pairs from the batch
- **Feed those features to the discriminator along with the image**
- **Feature values will differ b/w diverse and non-diverse batches**
  - Thus, Discriminator will rely on those features for classification
- **This in turn,**
  - Will force the Generator to match those feature values with the real data
  - Will generate diverse batches

Basic (Heuristic) Solutions

- Mini-Batch GANs
- **Supervision with labels**

Supervision with Labels

- Label information of the real data might help



- Empirically generates much better samples

Wasserstein GAN - WGAN

- Pitfalls of GAN
  - No guarantee to equilibrium
  - The discriminator only gives 0 or 1 but cannot describe how good or bad the image is

  https://github.com/soumith/ganhacks

- WGAN
  - Wasserstein distance between two data distributions
  - The discriminator gives a continuous evaluation describe how good or bad the image is

Arjovsky, Martin, and Léon Bottou. "Towards principled methods for training generative adversarial networks." *arXiv preprint arXiv:1701.04862* (2017).
M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," arXiv preprint arXiv:1701.07875, 2017.
I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," arXiv preprint arXiv:1704.00028, 2017.

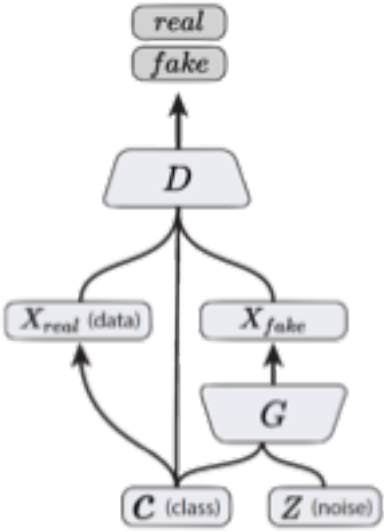
Conditional GANs

MNIST digits generated conditioned on their class label.

Figure 2 in the original paper.

Conditional GANs

- Simple modification to the original GAN framework that conditions the model on *additional information* for better multi-modal learning.
- Lends to many practical applications of GANs when we have explicit *supervision* available.

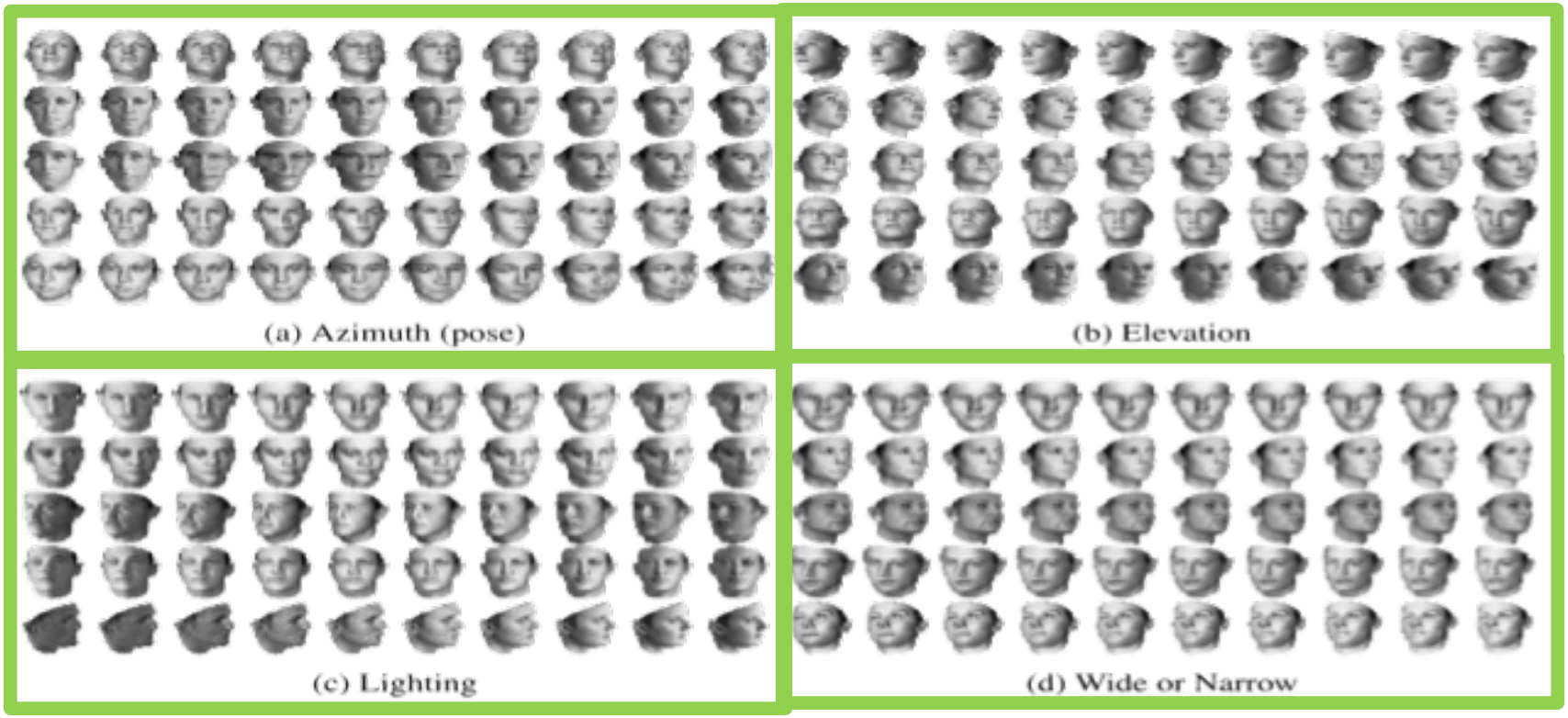


Conditional GAN
(Mirza & Osindero, 2014)

Image Credit: Figure 2 in Odena, A., Olah, C. and Shlens, J., 2016. Conditional image synthesis with auxiliary classifier GANs. *arXiv preprint arXiv:1610.09585*.

The Cool Stuff…

3D Faces



Cool Stuff (contd.)

3D Chairs

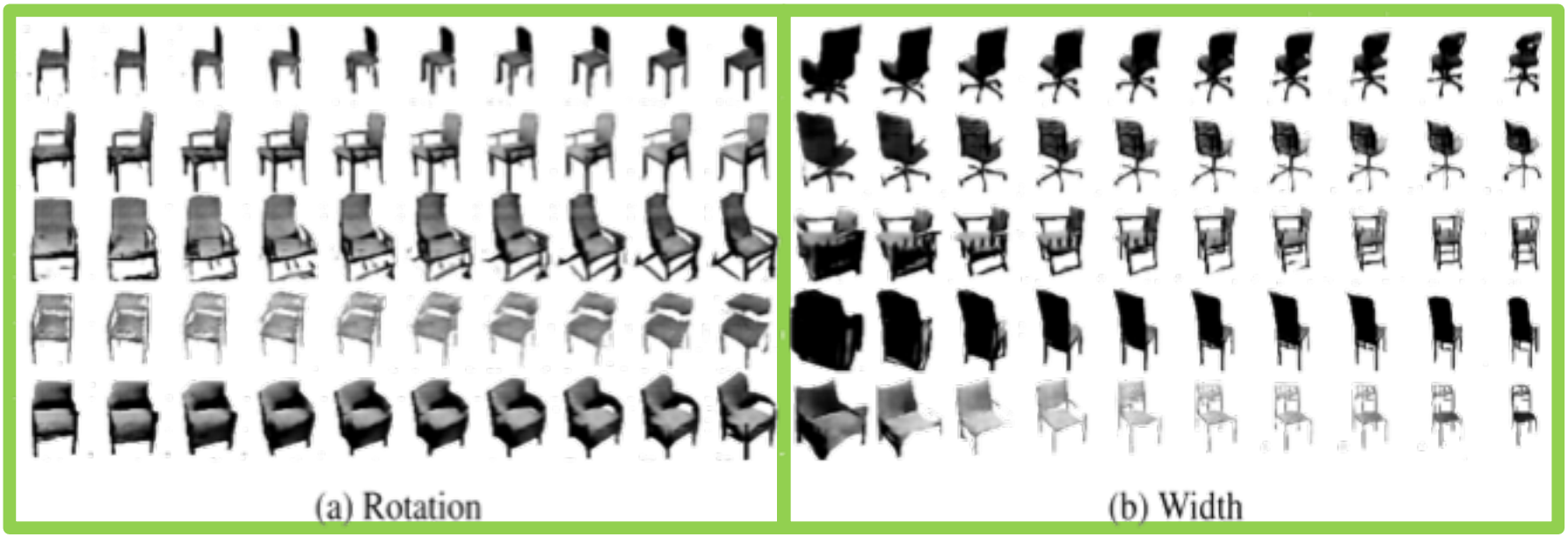


Image-to-Image Translation

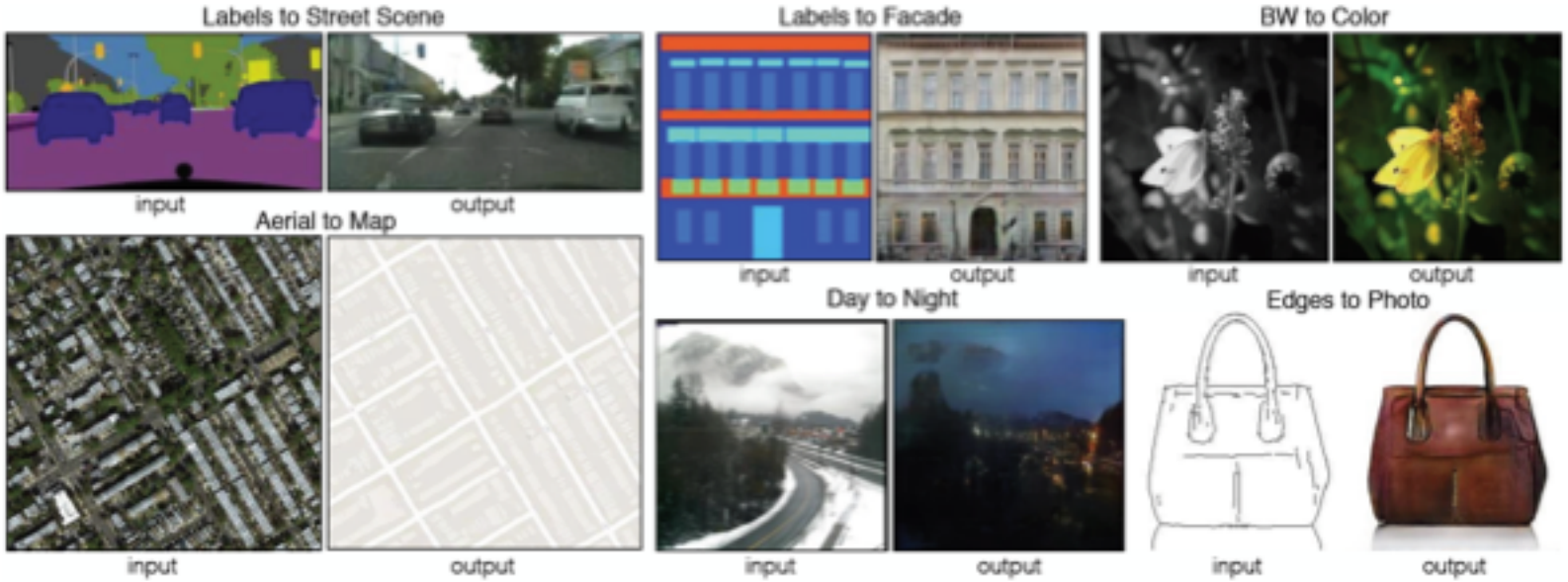


Figure 1 in the original paper.

[Link to an interactive demo of this paper]

Image-to-Image Translation

- Architecture: *DCGAN*-based architecture
- Training is conditioned on the images from the source domain.
- Conditional GANs provide an effective way to handle many complex domains without worrying about designing *structured loss* functions explicitly.

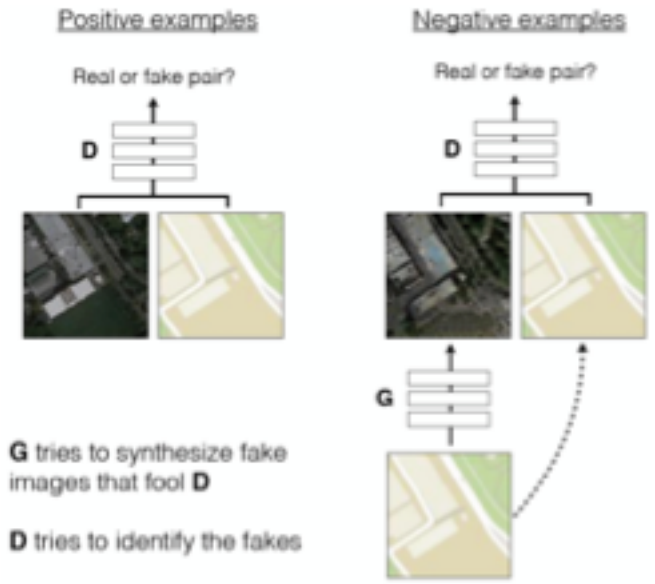


Figure 2 in the original paper.

Text-to-Image Synthesis

Motivation

Given a text description, generate images closely associated.

Uses a conditional GAN with the generator and discriminator being condition on “dense” text embedding.

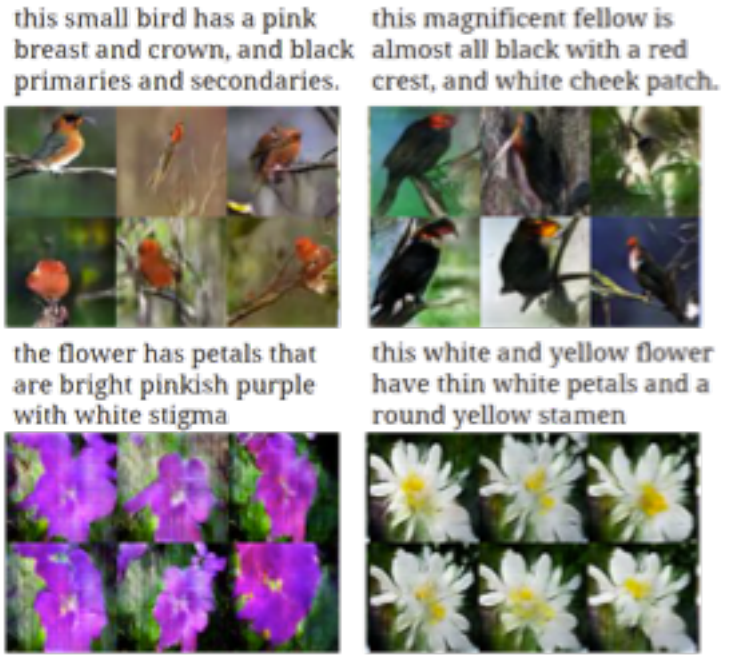


Figure 1 in the original paper.

Text-to-Image Synthesis

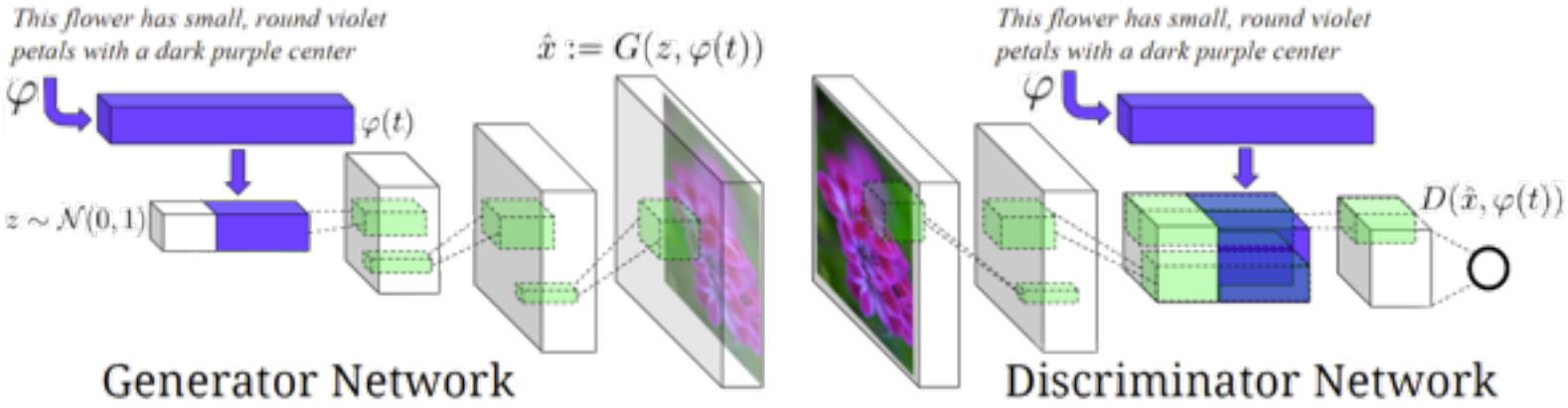


Figure 2 in the original paper.

Positive Example:
Real Image, Right Text

Negative Examples:
Real Image, Wrong Text
Fake Image, Right Text

Face Aging with Conditional GANs

- Differentiating Feature: Uses an *Identity Preservation Optimization* using an auxiliary network to get a better approximation of the latent code (z*) for an input image.
- Latent code is then conditioned on a discrete (one-hot) embedding of age categories.

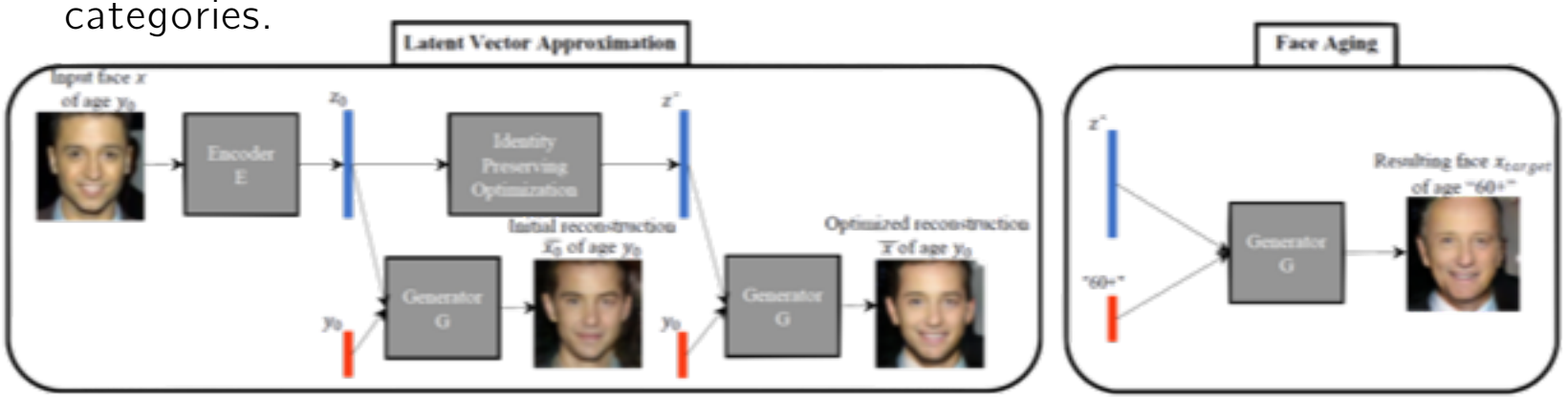


Figure 1 in the original paper.

Face Aging with Conditional GANs

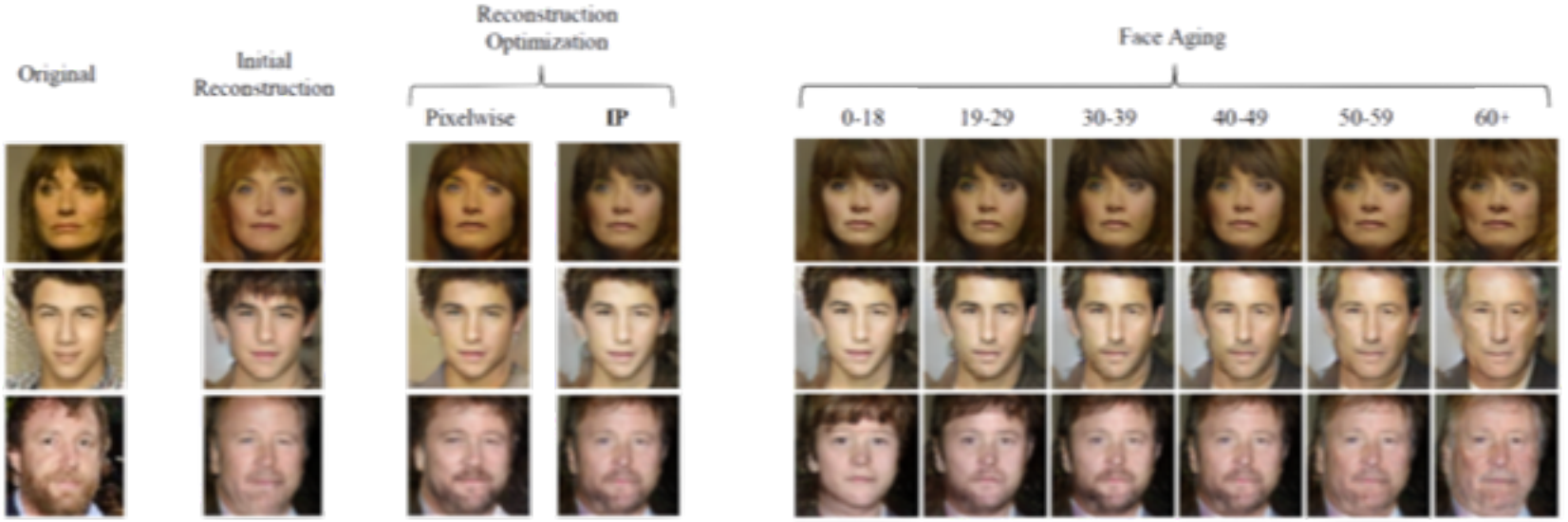


Figure 3 in the original paper.

Coupled GAN

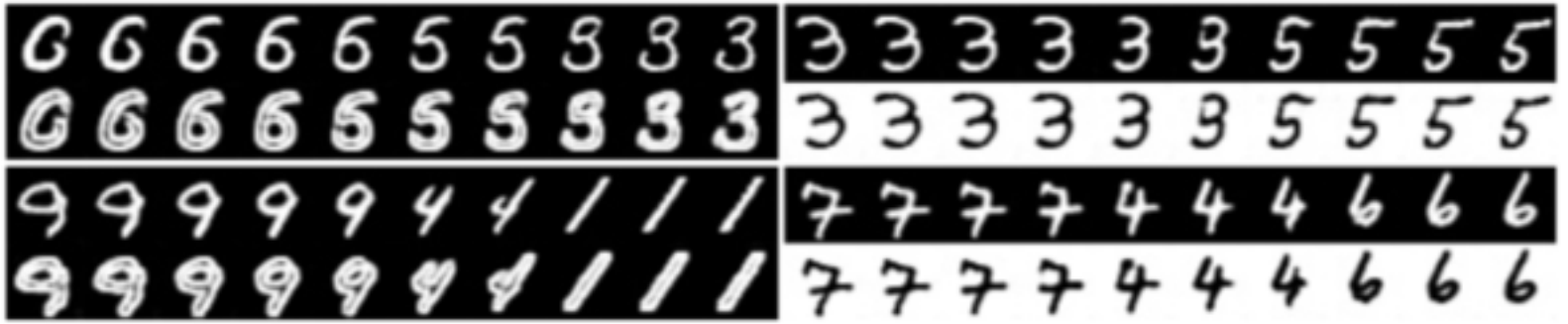


Figure 2 in the original paper.

- Learning a *joint distribution* of *multi-domain* images.
- Using GANs to learn the joint distribution with samples drawn from the marginal distributions.
- Direct applications in domain adaptation and image translation.

Liu, Ming-Yu, and Oncel Tuzel. "Coupled generative adversarial networks." *NIPS 2016*

Coupled GANs

- Architecture

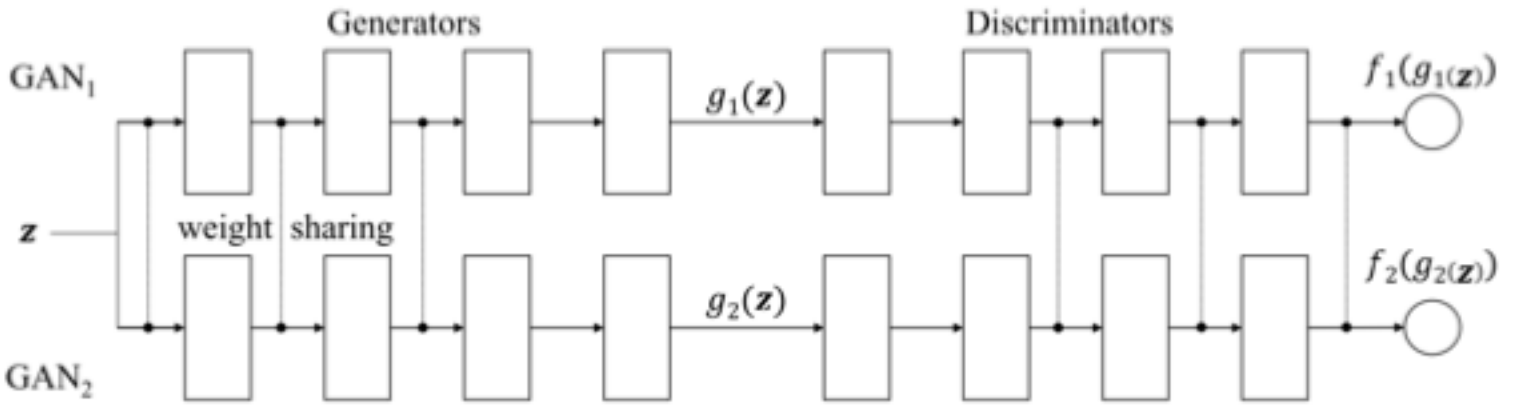


Figure 1 of the original paper.

Weight-sharing constraints the network to learn a *joint distribution* without corresponding supervision.

Liu, Ming-Yu, and Oncel Tuzel. "Coupled generative adversarial networks." *NIPS 2016*

Coupled GANs

- Some examples of generating facial images across different feature domains.
- Corresponding images in a column are generate from the same latent code z

Figure 4 in the original paper.

Liu, Ming-Yu, and Oncel Tuzel. "Coupled generative adversarial networks." *NIPS 2016*

Laplacian Pyramid of Adversarial Networks

Figure 1 in the original paper. (Edited for simplicity)

- Based on the Laplacian Pyramid representation of images. (1983)
- Generate high resolution (dimension) images by using a hierarchical system of GANs
- Iteratively increase image resolution and quality.

Laplacian Pyramid of Adversarial Networks

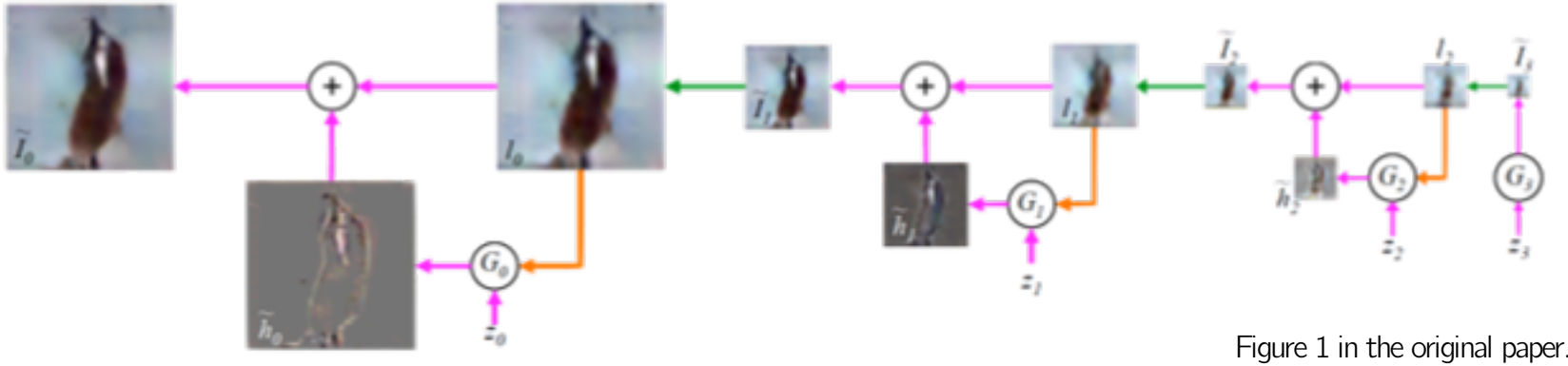


Figure 1 in the original paper.

Image Generation using a LAPGAN

- Generator G_3 generates the base image I_3 from random noise input z_3.
- Generators (G_2, G_1, G_0) iteratively generate the *difference image* $(\hat{h})$ conditioned on previous small image (l).
- This *difference image* is added to an up-scaled version of previous smaller image.

Laplacian Pyramid of Adversarial Networks

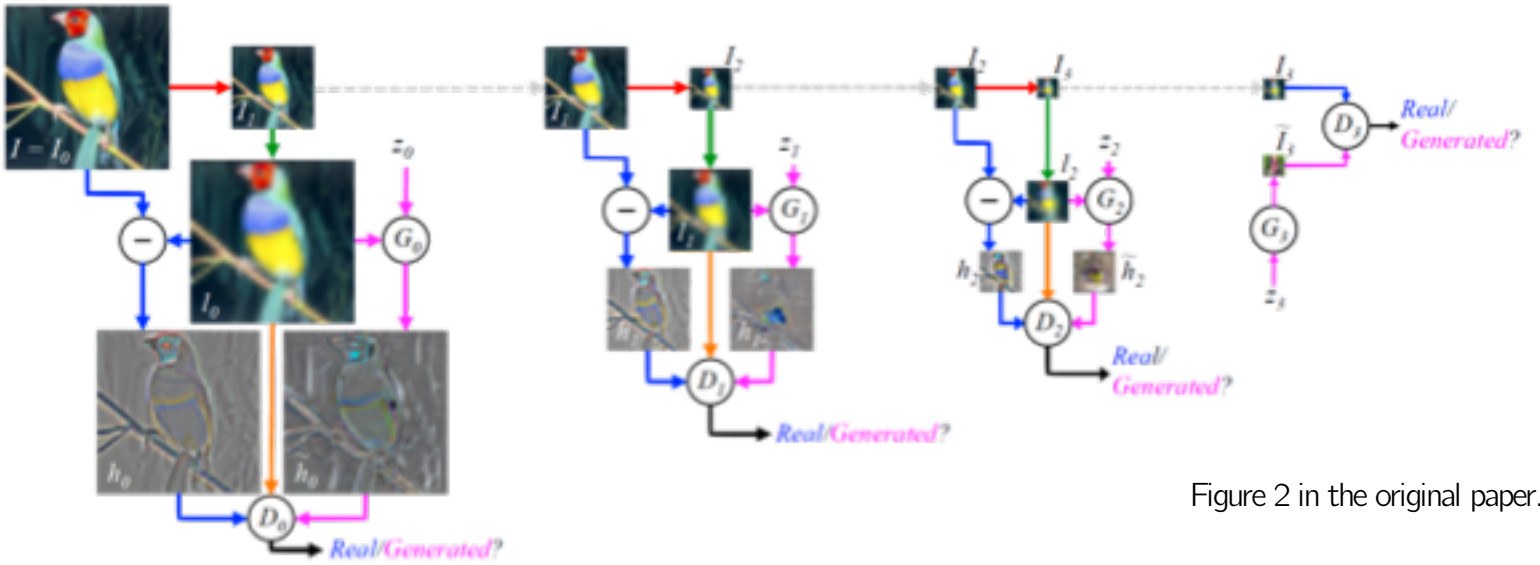


Figure 2 in the original paper.

Training Procedure:

Models at each level are trained independently to learn the required representation.

Summary

- GANs are generative models that are implemented using two stochastic neural network modules: **Generator** and **Discriminator**.
- **Generator** tries to generate samples from random noise as input
- **Discriminator** tries to distinguish the samples from Generator and samples from the real data distribution.
- Both networks are trained adversarially (in tandem) to fool the other component. In this process, both models become better at their respective tasks.

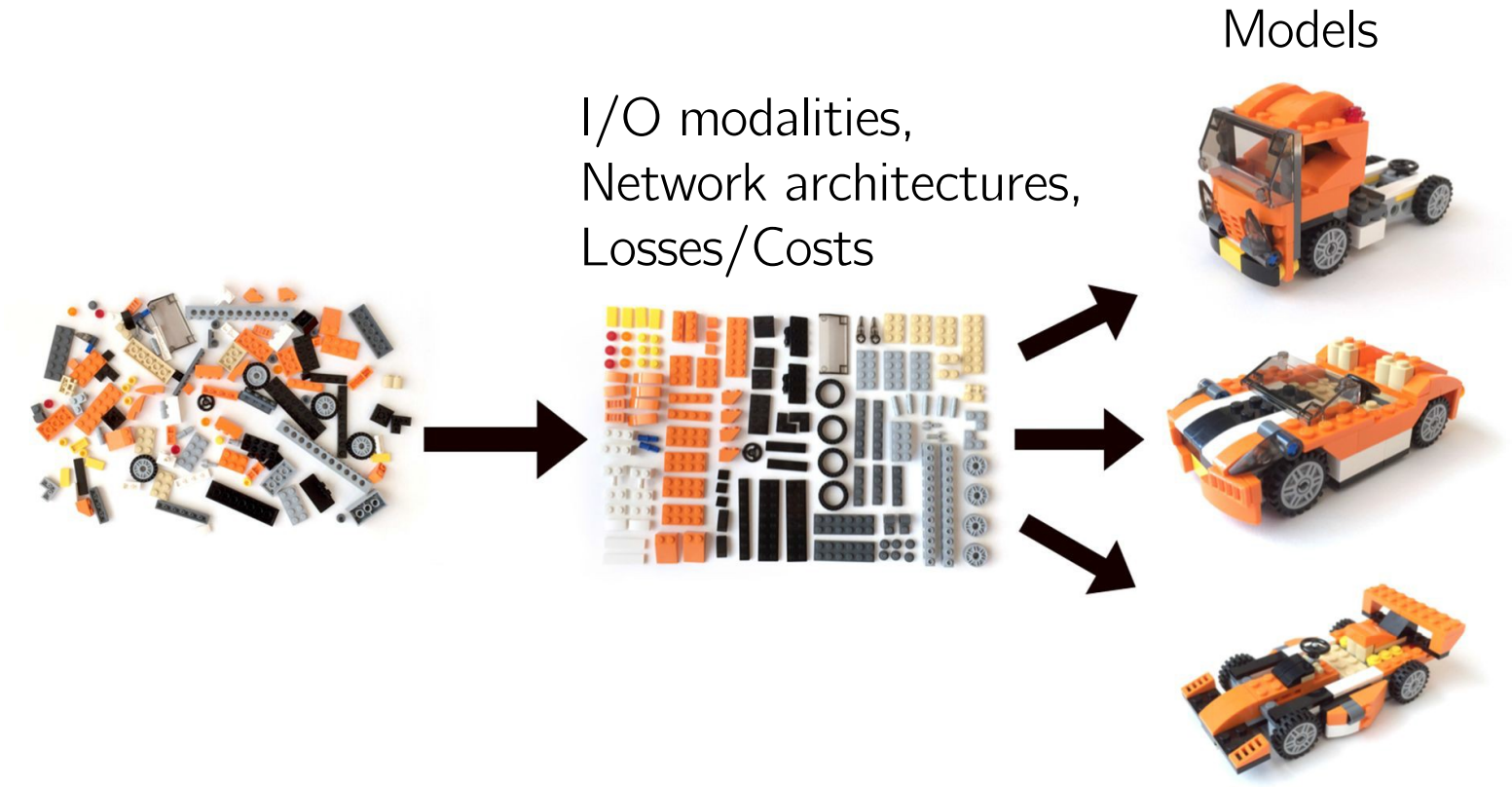
- Active areas of research:
  - Better loss functions (WGAN, LSGAN,...)
  - Conditional GANs and all kinds of applications

https://github.com/hindupuravinash/the-gan-zoo

Why use GANs for Generation?

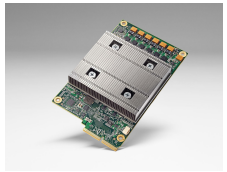
- Can be trained using back-propagation for Neural Network based Generator/Discriminator functions.
- Sharper images can be generated.
- Faster to sample from the model distribution: *single* forward pass generates a *single* sample.

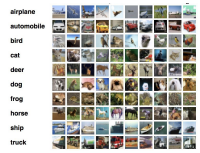
Deep Learning Building Blocks

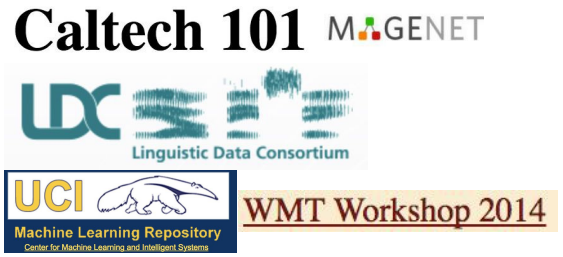


Nagel, Wolfram. Multiscreen UX Design: Developing for a Multitude of Devices. Morgan Kaufmann, 2015.

Deep Learning: Zooming Out







Deep Learning: Zooming In

Non-linearities

ReLU

Sigmoid

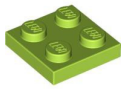
Tanh

GRU

LSTM

Linear…?

Connectivity



Fully connected

Convolution

Recurrent

Recursive

Skip/Residual

Random…?

Optimizer

SGD

Momentum

RMSProp

Adagrad

Adam

…

Loss

Cross Entropy

Adversarial

Variational

Max. Likelihood

Sparse

L2

HyperParameters

Learning Rate

Layer Size and #

Batch Size

Dropout

Weight initialization

Data augmentation

Gradient clipping

Weight decay

Momentum

…

Finding your blocks

INPUTS/ OUTPUTS

ARCHITECTURE

LOSS

Image pixels / Class labels

Text sequences

Audio waveforms

Sets of images/texts (no labels)

Convolutions

Recurrent (over space/time)

Attention

Discrete: softmax cross entropy (with L2 reguralization)

Continuous: Gaussian (mixture) likelihood

Adversarial loss

Thanks!

Reading List (general)

- Books:
  - http://www.deeplearningbook.org/
  - http://neuralnetworksanddeeplearning.com/
- Courses:
  - http://cs231n.stanford.edu/
  - http://www.cs.toronto.edu/~rgrosse/csc321/
  - http://web.stanford.edu/class/cs224n/
- Guides to deep learning:
  - http://yerevann.com/a-guide-to-deep-learning/
  - http://ufldl.stanford.edu/tutorial/
  - https://github.com/terryum/awesome-deep-learning-papers

Reading List (RNNs)

- R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013
- S. Hochreiter, and J. Schmidhuber, Long short-term memory, Neural computation, 1997 9(8), pp.1735-1780
- F.A. Gers, and J. Schmidhuber, Recurrent nets that time and count, IJCNN 2000
- K. Greff , R.K. Srivastava, J. Koutník, B.R. Steunebrink, and J. Schmidhuber, LSTM: A search space odyssey, IEEE transactions on neural networks and learning systems, 2016
- K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014
- R. Jozefowicz, W. Zaremba, and I. Sutskever, An empirical exploration of recurrent network architectures, JMLR 2015
- Seq2Seq ICML 2017 Tutorial (Vinyals & Jaitly)
- Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation.” EMNLP’15.
- Andrychowicz, Marcin, and Karol Kurach. "Learning efficient algorithms with hierarchical attentive memory." *arXiv preprint arXiv:1602.03218* (2016).
- Xu, Kelvin, et al. “Show, attend and tell: Neural image caption generation with visual attention.” ICML 2015

Reading List (GANs)

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative adversarial nets, NIPS (2014).
- Goodfellow, Ian NIPS 2016 Tutorial: Generative Adversarial Networks, NIPS (2016).
- Radford, A., Metz, L. and Chintala, S., Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434. (2015).
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. Improved techniques for training gans, NIPS (2016).
- Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).
- Zhao, Junbo, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016).
- Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).
- Liu, Ming-Yu, and Oncel Tuzel. Coupled generative adversarial networks. NIPS (2016).
- Denton, E.L., Chintala, S. and Fergus, R., 2015. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. NIPS (2015)
- Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., & Courville, A. Adversarially learned inference, arXiv preprint arXiv:1606.00704 (2016).

Applications:

- Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004. (2016).
- Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. Generative adversarial text to image synthesis. JMLR (2016).
- Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). Face Aging With Conditional Generative Adversarial Networks. arXiv preprint arXiv:1702.01983.