Introduction to Deep Learning

Solution to Lecture Questions

1 – Introduction

What is the formal definition of Machine Learning?

A computer program that learns from experience E with respect to some tasks T and performance measure P, such that its performance at tasks in T, as measured by P, improves with experience E."

Does a validation / test set still make sense for unsupervised learning?

Yes, in order to find out if the findings (for example clustering) generalize well.

2 – Linear Regression

Give the objective function of linear regression

Mean-squared error (MSE):

Derive the closed form solution for linear regression

To find the minimum of the function $E(\boldsymbol{w})$, compute the gradient $\nabla_{\boldsymbol{w}} E(\boldsymbol{w})$:

$$\nabla_{\boldsymbol{w}} E_{\mathrm{LS}}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \frac{1}{2} (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) \tag{11}$$

$$= \nabla_{\boldsymbol{w}} \frac{1}{2} \left(\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y} \right) \tag{12}$$

$$= \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y} \tag{13}$$

Now set the gradient to zero and solve for $\boldsymbol{w}$ to obtain the minimizer [2]

$$\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - \boldsymbol{X}^T \boldsymbol{y} \stackrel{!}{=} 0 \tag{14}$$

This leads to the so-called normal equation of the least squares problem

$$\boldsymbol{w}^* = \underbrace{(\boldsymbol{X}^\mathsf{T} \boldsymbol{X})^{-1} \boldsymbol{X}^\mathsf{T}}_{=\boldsymbol{X}^\dagger} \boldsymbol{y} \tag{15}$$

For univariate linear regression, show that MLE = Least Squares Estimate given that the mean is modeled by the linear regression and the variance is fixed.

- We have **2 equivalent alternatives to maximizing the likelihood:**

$$\boxed{\boldsymbol{w}_{\boldsymbol{ML}}, \beta_M = \arg\max_{\boldsymbol{w},\beta} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta)} \text{ or } \boxed{\boldsymbol{w}_{\boldsymbol{ML}}, \beta_M = \arg\min_{\boldsymbol{w},\beta} E_{ML}} \text{ or } \boxed{\boldsymbol{w}_{\boldsymbol{ML}} = \arg\min_{\boldsymbol{w}} E_{LS}}$$

- Maximize likelihood function $\begin{cases} \text{of a single sample:} & p(y_i \mid f_w(\boldsymbol{x}_i), \beta) = \mathcal{N}(y_i \mid f_w(\boldsymbol{x}_i), \beta^{-1}) \\ \text{of the entire dataset:} & p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{i=1}^N p(y_i \mid f_w(\boldsymbol{x}_i), \beta) \end{cases}$

$$\boxed{\boldsymbol{w}_{\boldsymbol{ML}}, \beta_{ML} = \arg\max_{\boldsymbol{w},\beta} [\prod_{i=1}^N p(y_i \mid f_w(\boldsymbol{x}_i), \beta)]}$$

- Minimize maximum likelihood error function: $E_{ML}(\boldsymbol{w}, \beta) = -\ln p(y_i \mid f_w(\boldsymbol{x}_i), \beta)$

$$\boxed{\boldsymbol{w}_{\boldsymbol{ML}}, \beta_{ML} = \arg\min_{\boldsymbol{w},\beta} [-\ln p(y_i \mid f_w(\boldsymbol{x}_i), \beta)]}$$

$\boxed{\ln \prod_i x_i = \sum_i \ln x_i}$ (switches max-min!)

- Minimize least squares error function: $E_{LS}(w) = -\frac{1}{2} \sum_{i=1}^N (w^T \phi(x_i) - y_i)^2$

$$\boxed{\boldsymbol{w}_{\boldsymbol{ML}} = \arg\min_{\boldsymbol{w}} \left[-\frac{1}{2} \sum_{i=1}^N (w^T \phi(x_i) - y_i)^2 \right]}$$

Further simplify and β disappears

y

Give the equation for the binary cross-entropy loss

For binary classification (M=2):

Why should you use the analytical gradient over the numerical gradient?

Exact and fast vs approximate and slow

What is the difference between L1 and L2 regularization?

- L1 enforces sparse weights, L2 weights will take "all information into account"
- L2 enforces weights with similar values

L2 regularization	L1 regularization
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

Explain why dropout in a neural network acts as a regularizer (question from Stanford exam 2018).

Solution: There were several acceptable answers:

(1) Dropout is a form of model averaging. In particular, for a layer of H nodes, we sampling from 2H architectures, where we choose an arbitrary subset of the nodes to remain active. The weights learned are shared across all these models means that the various models are regularizing the other models.

(2) Dropout helps prevent feature co-adaptation, which has a regularizing effect.

(3) Dropout adds noise to the learning process, and training with noise in general has a regularizing effect.

(4) Dropout leads to more sparsity in the hidden units, which has a regularizing effect. (Note that in one of the lecture videos, this was phrased as dropout "shrinking the weights" or "spreading out the weights". We will also accept this phrasing.)

What is regularization?

Any strategy that aims to lower the validation error by increasing the training error

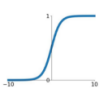
3 - Introduction to Neural Networks

Why activation functions?

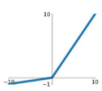
Activation functions introduce non-linear properties to the NN. They allow non-linear mappings from inputs to outputs to be learnt. Without them, the models would remain linear from layer to layer, and we couldn't model complex relationships. ("Input times weights , add Bias and Activate")

Name and give formulas for seven different activation functions

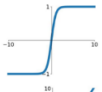
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

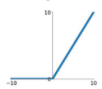


tanh
$\tanh(x)$

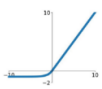


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Softmax:

Why organize a neural network into layers?

- Fast to compute with matrix operations
- Layers are modular and can be easily stacked for distinct architectures
- Architectures can be easily mixed and compared
- Enables weight sharing in CNN layers

Why should you use a computational graph?

- Can easily modularize complex functions
- Language agnostic -> create in python, execute in C

What happens if there are multiple outputs in a compute node?

I believe Prof. Leal-Taixe said that the outputs should be averaged yes!

What happens if there are loops in the graph?

There cannot be loops in the graph (acyclic)

Why backpropagation and not forward propagation of gradients?

The purpose of backpropagation is to efficiently calculate the weight updates in order to improve the network. First, the forward propagation must evaluate the cost for the given set of weights. Only after that is calculating the gradients in the backpropagation step meaningful: in order to update the weights as to decrease the loss (negative gradient direction). After that we must propagate again, and so recursively.

4 - Backpropagation

Derive the gradients for a two-layer neural network with cross-entropy loss

Derive the sigmoid function

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Derivative of the Sigmoid Function

src: https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

Draw the computational graph for two inputs, no hidden layer, three outputs and squared error

What does w_{abc} denote?

- a – layer
- b – neuron
- c – weight

Draw the computational graph for one hidden layer

5 - Optimization

How is a convex function defined?

- If lines/planes between any two points lie above or on the graph of the function

Give the equation for a single parameter update on a single training example with SGD

-

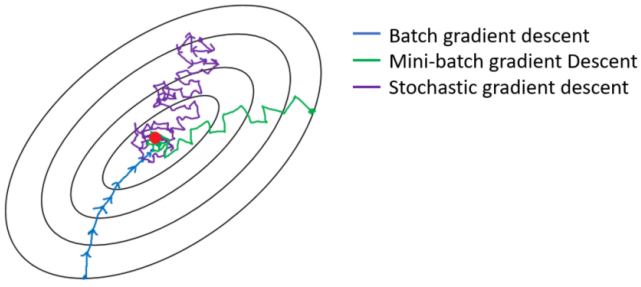
What is the difference between iteration and epoch?

- Iteration: One training step (update of the parameters) / one pass through the mini-batch
- Epoch: One complete pass through training set

- If dataset is small and fits into the memory, then batch gradient descent would mean 1 iteration = 1 epoch

What are the problems of SGD?

- Gradient is scaled equally across all dimensions
  - cannot independently scale directions
  - need to have conservative min learning rate to avoid divergence
  - slower than necessary
  - ravines
- Finding a good learning rate is difficult slow convergence or fluctuation around minimum
- Gets trapped in local minima



What are the differences between batch, stochastic and mini-batch gradient descent?

	Batch GD	Mini-batch GD	Stochastic GD
At each step, compute gradient of cost function ...	on entire training set	on multiple examples	on a single example
Redundant computations? (for large datasets)	Yes	Reduced redundancy	No
Variance of updates:		Reduced variance → more stable convergence Makes use of optimized matrix optimizations	high variance → objective function fluctuates heavily → can lead to basins (local minima) and prevent convergence → needs learning rate decay

Give the update equation of SGD with Momentum

Why do we need momentum?

- Dampens out oscillation: by exponentially weighting average of gradients
- Speeds-up convergence: accumulation of velocity will speed up the movement towards goal

How does Nesterov's Momentum work?

- Improved version of Momentum with velocity (computes gradient at instead of)

What does RMSProp do? Give the update equation

- Divides the learning rate by an exponentially-decaying average of squared gradients

$$s_{k+1} = \beta\, s_k + (1-\beta)(\nabla_\theta L)^2$$

$$\theta_{k+1} = \theta_k - \frac{\alpha}{\sqrt{s_{k+1}} + \epsilon} \nabla_\theta L$$

What is the advantage of RMSProp?

- dampens oscillations for high-variance directions less likely to diverge
- learning rate can be increased

What does Adam do? Give the update equations

- Momentum + RMSProp

  (1st momentum: mean of gradients)

  (2nd momentum: variance of gradients)

  (parameter update)

What does bias correction mean for Adam?

- m and v are initialized with zero bias towards zero bias-corrected moment updates:

  and

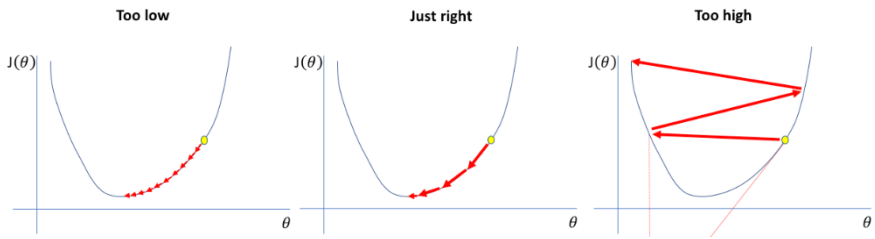
What is the advantage of Adam?

- Combines momentum and RMSprop first and second order momentum
- "Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface"

What does AdaGrad do? Why is AdaGrad not used very often?

- Adapt the learning rate of all model parameters
- It accumulates gradients from the beginning
- Problem: accumulation of the squared gradients in the denominator causes learning rate to shrink until becoming infinitesimally small

Draw a very high learning rate, high learning rate, good learning rate and low learning rate



I think here they would like to see the 3 different curves of learning rates as seen in lecture 6 depending on # of epochs.

What is the difference between a gradient and a derivative?

- Derivative – the rate of change of a function at a point (scalar value)
- Gradient – the vector pointing to the direction of the steepest ascent of a curve (vector)
- A gradient is the multi-variable generalization of the derivative

True or false: the Hessian Matrix is the transposed Jacobian Matrix of the gradient?t

- True

~~I think it's false! The Hessian is "the Jacobian of the Jacobian" (2nd order derivatives), not just the transpose.~~

Follow-up: From Wikipedia: `The Hessian matrix can be considered related to the Jacobian matrix by H(f(x))=J(▽f(x))^T` So I think this is also true.

You're right. It's the transposed Jacobian of the **gradient (I didn't catch the word "gradient" in the question).**

Explain Newton's method

- A second-order method for computing the roots of a function f in an iterative manner
- The method first takes a point , then updates this value in each iteration until the value converges. The converged result is a root of .
- It approximates the function by a second-order Taylor series expansion
- Requires computing the Inverse of the Hessian matrix

Can you apply Newton's method for linear regression? What do you get as a result?

- If the loss function is quadratic (as in Least Squares Loss for linear regression) the optimal solution can be found in a single iteration.
- The learning rate is replaced by the inverse of the Hessian (a representation of the curvature of the graph)

Could someone explain what the difference between a "linear system" and linear regression is?

For a dataset with 10.000 examples and very low redundancy, is Adam a good choice as an optimizer. Why/Why not?

No. For small datasets with low redundancy, full-batch methods like L-BFGS can be applied and are generally preferred in this case.

Source: Geoff Hinton's coursera lessen https://youtu.be/defQQqkXEfE?t=456

Why can't you simply use Newton's method for Deep Learning?

- Requires computing the inverse of the Hessian matrix which is computationally infeasable for large datasets. Hessian matrix H of a NN with n params has:
  - elements in H
  - computational complexity is
- Also, Newton variations don't work well for mini-batches (due to stochasticity)

6 Optimization 2

Give two ways of decaying the learning rate

Exponential:

Manually

step decay: **= - t***

What happens if the learning rate is too high? What if too low?

- too high:
  - training error increases
  - or oscillates around local minimum
  - overshooting minima
- too low:
  - very slow convergence

What problem(s) will result from using a learning rate that's too high? How would you detect these problems?

Cost function does not converge to an optimal solution and can even diverge. To detect, look at the costs after each iteration (plot the cost function v.s. the number of iterations). If the cost oscillates wildly, the learning rate is too high. For batch gradient descent, if the cost increases, the learning rate is too high.

What is a saddle point? What is the advantage/disadvantage of SGD in dealing with saddle points?

Solution: Saddle point - The gradient is zero, but it is neither a local minima nor a local maxima. Also accepted - the gradient is zero and the function has a local maximum in one direction, but a local minimum in another direction. SGD has noisier updates and can help escape from a saddle point

What are training, validation and test set for?

- Training set:
  - for training the neural network (learning parameters)
- Validation set:
  - for hyperparameter optimization
  - to check generalization progress
- Test set:
  - for final evaluation
  - only for the very end
  - never touch during development or training

Name hyperparameters

- In order of importance:
  - Learning rate
  - # hidden units
  - Mini batch size
  - Momentum turn
  - # layers
  - Learning rate decay

When should you use cross validation?

- When data set is very small and/or model has low training times

How do you do cross validation?

- Partition data into k subsets, train on k-1 and evaluate performance on the remaining subset
- Compute mean and standard deviation of the different performance scores

To avoid overfitting, you can

increase the size of the network.
use data augmentation.
decrease the learning rate.
stop training earlier.

7 – Training NNs

Write down the Softmax and its MLE loss function

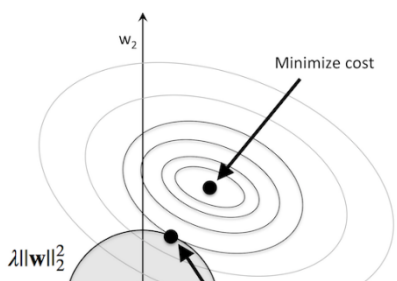
- -log (softmax output for target)

- **Softmax:**

Compare L1 to L2 loss as output losses

- L1 enforces sparse weights, L2 weights will take "all information into account"
- L2 enforces weights with similar values

L2 Loss (Ridge)	L1 Loss (Lasso)
Sum of squared differences	Sum of absolute differences
Prone to outliers	Robust to outliers
Always one solution	Possibly many solutions



Optimum is the mean	Optimum is the median
Keeps all weights ("takes all info into account") Able to learn complex models	Discards unimportant weights (enforces sparse solutions) Generates simple models, has built-in feature selection

Give the equations for 4 different output losses

- L2 Loss sum_iˆn (y_i - f(x_i))ˆ2
- L1 Loss sum_iˆn |y_i - f(x_i)|
- Xent softmax loss -log (softmax output for target)
- Hinge loss sum_j != y_i max(0, s_j - s_y_i + 1) -> overall loss is averaged over samples

What is the difference between softmax loss and hinge loss?

- Softmax loss always wants to improve
- Hinge loss saturates
- In general: classification problems often use SoftMax loss since it always attempts to improve the classifier. Hinge loss on the other hand is used for support vector machines.

What are the advantages and disadvantages of the distinct activation function?

Sigmoid	Tanh	ReLU	LeakyReLU	Parametric ReLU	Maxout units
+ strong gradient around 0 + active region for gradient descent - vanishing gradient problem - output always positive inefficient updates	+ zero-centered - vanishing gradient problem - strong gradient only around zero	+ fast convergence + doesn't saturate - zero input/output kills gradient + large and consistent gradients	+ fast convergence + doesn't saturate + doesn't lead to dead units + large and consistent gradients - slope parameter needs to be tuned	+ fast convergence + doesn't saturate + doesn't lead to dead units + large and consistent gradients - slope parameter needs to be trained and bounded	+ generalization of ReLU + doesn't saturate + doesn't lead to dead units + linear regimes - increase of the number of params

- We want our input to have the same variance as our score (before activation function)

Give a quick guide to activation functions

- ReLU is standard choice (but be careful with learning rate and dead units)
- If ReLU is giving problems. Try Leaky ReLU, Parametric ReLU or Maxout. Do not use sigmoid
- RNNs require tanh or similar
- Deep networks cannot use sigmoid nor tanh due to the vanishing gradient problem

Name a method used in data preprocessing

- For images, subtract the mean image
- data augmentation
- image brightness / contrast / ..

8 - Training NNs 2

Why shouldn't you just initialize the weights with zero?

- Hidden units all compute the same Gradients all the same Need to break symmetry somehow

Why shouldn't you just randomly initialize the weights with a small standard deviation?

- Activations quickly become zero, even after only 10 layers
- Multiplying independent Gaussians with small standard deviation with each other pushes the values towards zero
- Gradients vanish because inputs to layers are close to zero

What does the Xavier initialization do?

- It initializes weights by sampling a zero-mean Gaussian distribution with variance 1/n (with n = number of neurons in layer)

When does Xavier fail? What can you do?

- Fails for ReLU, more and more activations become zero with increasing layers
- Double variance to 2/n

Where in the architecture should the BN layer be applied?

- After the weights (fully connected or convolutional) and before the non-linearity

Why do you need gamma and beta for BN?

- Enable a layer to undo the BN
- (Unit Gaussians everywhere might not be best for the network)

How does BN work at test time?

- Compute mean and variance by running an exponentially weighted average across training mini-batches

What are the benefits of Batch Normalization?

- Very deep nets are much easier to train because of more stable gradients
- Much larger range of hyperparameters works similarly
- Works with (almost) arbitrary weight initialization
- Works with many, especially saturating activation functions
- Reduces internal covariate shift

Add weight decay to E

- E = E + lambda delta^T delta

What are the benefits of weight decay?

- Penalizes large weights
- Improves generalization

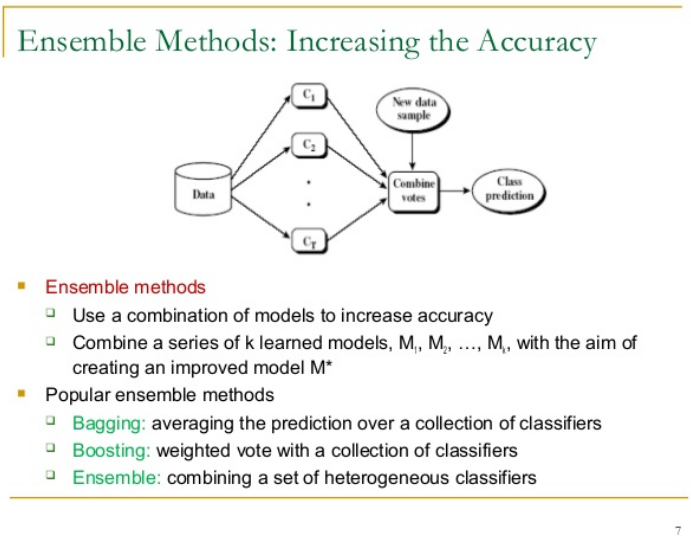
Explain methods of data augmentation for images. If necessary, also explain what to do during test time

- (Random) cropping or fixed set of crops if during test time
- Flipping the image
- Random brightness and contrast changes

Name regularization techniques

- Weight decay
- Dropout
- Data augmentation
- Early stopping
- Bagging (different learning algorithm, different objective function etc.)

What is a condition for an error decrease caused by ensemble methods?



- Uncorrelated errors -> error will decrease linearly with the ensemble size

What is the intuition behind dropout?

Central: reducing co-adaptation between neurons

Causes redundant representations:

- Randomly removing neurons avoids highly specialized neurons with high outgoing weights
- Forces more distributed representations and more balanced weights
- -> Robust to small changes in the input

Can be considered as an ensemble of networks:

- Efficient bagging method with parameter sharing
- Dropout implicitly creates ensemble of various networks, where each network is only trained on one mini-batch
- Each model in the ensemble makes an error
- When taking the average of the predictions, the errors are also averaged
- For uncorrelated errors, this linearly decreases the variance of the error with the ensemble size

- Errors are not uncorrelated, but also not completely correlated because of weight inits and random mini-batches

What is the disadvantage of dropout?

Reduces the effective capacity of a model -> larger models need more training time

cost function is not so well defined anymore

What is the weight scaling inference rule?

Scale layer's output with dropout rate at test time to keep the expected total activation level the same

Name two architectures in that use weight sharing.

1. CNNs (filters)
2. RNNs (hidden state)

9 – CNNs

What are the three assumptions of CNNs?

- Compositionally: hierarchy of local features
  - Nearby pixels are correlated
  - Next layer combines local features to more complex and more global features
- Weight sharing -> interesting features all over the place
- Translational invariance

What are the advantages of Convolutional Layers compared to Fully Connected Layers?

By assuming compositional / hiarcharchy of local features and translational invariance, the number of parameters can greatly be reduced. -> much less training samples needed

What are the important parameters of a conv layer?

- Input size N
- Spatial filter extent F
- Number of filters K
-
- Padding P
- Stride S

What are the important parameters of a pooling layer?

- Input size N
- Spatial filter extent F
- Padding P (although padding makes no sense according to the lecture)
- Stride S

Give the equation for the number of neurons in a conv layer's feature map

((N + 2*P - F) / S) + 1

Give the equation for the number of parameters in a conv layer

F * F * D_in * K + K

Give the equation for the number of operations needed for a forward pass of a conv layer

Number of neurons * number of parameters

Why do you need padding?

Otherwise:

- Sizes get small too quickly
- Corner pixel is only used once
- Adds strong non-linearity without learnable parameters
- Max-pooling is hard to generate by normal neurons with linear input and non-linear activation function

What is the difference between valid and same padding?

Same padding keeps the input and output size=width*height constant, if stride = 1 P = (F - 1) / 2 Valid uses no padding

How can you backprop through CNN layers?

---

10 - CNNs 2

What architectures came after AlexNet?

- (ZFNet)
- VGG
- GoogleNet
- Resnet

What were the improvements of LeNet? How many parameters did it have?

- Actual use of conv layers
- Reduce spatial filter extent while increasing number of filters
- Use average pooling
- 60K parameters

What were the improvements of AlexNet? How many parameters did it have?

- Use max pooling
- Use ReLUs
- Use Dropout

- Use same padding
- 60M parameters

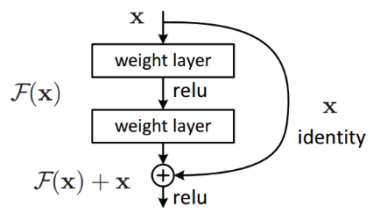
What were the improvements of VGG? How many parameters did it have?

- Striving for simplicity
- Smaller filters but more layers
- Replace larger filters with stack of smaller ones (7x7 -> 3 3x3 filters)
- Number of filters multiplied by 2 after a pooling layer
- Analyzed a lot of architectures
- 138M parameters

What were the improvements of Resnet?

- Solves the degradation problem (more layers give larger training and test error)
- Makes it easy for the residual block to learn a linear mapping, even with great depth
- Each block only learns a deviation from the linear mapping
- Skip connections: output = f(Wx + b + input)
- Use same padding, because we need same dimensions for the addition

Draw the concept of a Resnet block



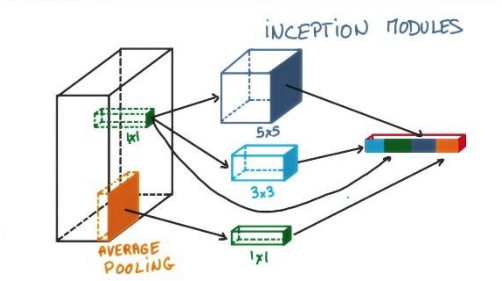
Why do we need 1x1 convolutions?

- To shrink the number of channels
- Further adds a non-linearity
- Combines feature maps for each pixel individually
- Kind of selecting features to be processed before an expensive convolution

What were the improvements of GoogleNet? How many parameters did it have?

- Use multiple filter sizes per inception module
- Multipath -> process parts of features separately
- 1x1 convolutions before expensive convolutions and after max pooling to reduce the share of the max pooling layer in the output
- Concatenate the feature maps of each path
- Drastically reduces the number of parameters (5M)

Draw an Inception module



What does transfer learning do?

- Re-use pretrained network and weights for a similar task

When does transfer learning make sense? (From P1 to P2)

- When task P1 and P2 have the same input (for example an RGB image)
- When you have more data for task P1 than for task P2
- When the low-level features of P1 could be useful to learn P2

11 RNNs

When do we need RNNs?

- When we have a variable input length and/or variable output length

Give the equations for a basic RNN

A_t = f(theta_c A_t-1 + theta_x x_t) h_t = g(theta_h A_t)

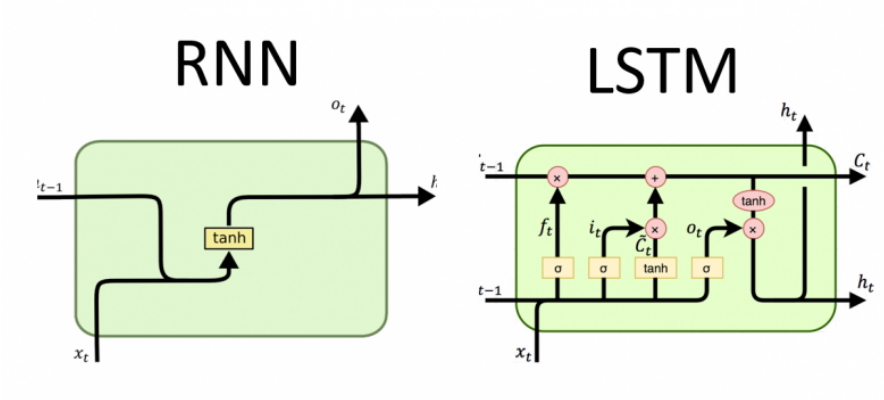
How can you backprop through a RNN?

Unroll it as a feedforward net all the way to t=0 -> weights will be the same in each step -> add the derivatives at different times for each weight

What is the problem of the basic RNN?

- Small weights / eigenvalues < 1 -> vanishing gradient
- Large weights / eigenvalues > 1 -> exploding gradient

Draw / compare RNN to LSTM cell



What are the gates in the LSTM for?

- Forget gate - decides when to erase the cell state
- Input gate - decides which values will be updated
- Output gate - decides which values will be outputted

Why does the LSTM work better than the RNN?

The addition instead of multiplication of new information and the lack of activation functions in the cell state flow create a highway for the gradient to flow less vanishing gradients

Write down the LSTM equations

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\theta_{xf}\mathbf{x}_t + \theta_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\theta_{xi}\mathbf{x}_t + \theta_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\theta_{xo}\mathbf{x}_t + \theta_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\widetilde{\mathbf{C}}_t &= \tanh(\theta_{xc}\mathbf{x}_t + \theta_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{C}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{C}_t)
\end{aligned}
$$

Appendix

Why does DL work so much better?

- Hierarchy of parameterized non-linear processing units are fundamentally better probabilistic model/prior for real-world data
- Automatic feature extraction and blurring artificial distinction between feature extraction and classification
- End-to-end learning in homogeneous architecture

Which of the following models can be used in unsupervised learning?

Autoencoder
PCA
K-means
Linear regression.