HANDS-ON AI I

Tricks of the Trade

Sohvi Luukkonen

Institute for Machine Learning

JOHANNES KEPLER UNIVERSITY LINZ

Institute for Machine Learning

Copyright Statement

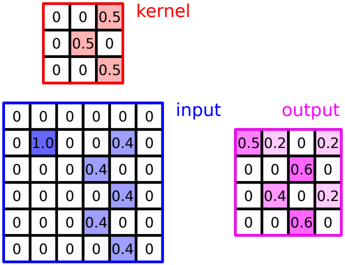
Exam

- Exam is on **February 5, 2023, 8:30**.
- Content: Everything.
- Please **register in KUSSS**.
- Please read the necessary information on Moodle.

Content of Unit 7

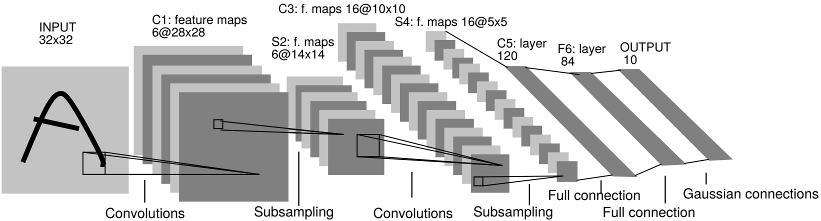
- Recap of last lecture:
  - Image Convolution
  - Convolutional Neural Networks
- Secret ingredients to make networks work well
- What's the catch?

Recap: Image Convolution



- Images are extremely **high-dimensional**:
  - 250 x 250 pixels x 3 color channels = 187.5k dimensions
- Pixels near each other are **highly correlated**.
- Interesting parts: pixels **different from their neighbors** (edges, corners). → Can be found with **convolutions**!
- Demo: `https://setosa.io/ev/image-kernels/`

Recap: Convolutional Neural Networks

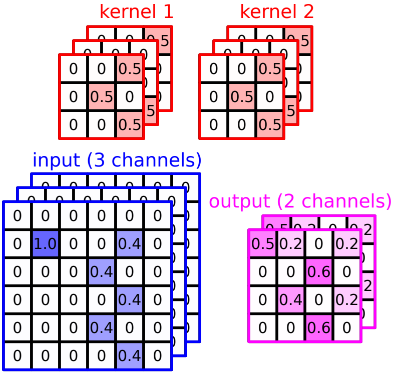


LeCun et al. (1998): Gradient-Based Learning Applied to Document Recognition

- Basic Convolutional Neural Networks (CNNs):
  - **Convolutional layers** to find local features.
  - **Pooling layers** to compress data.
  - **Fully-connected layers** to fuse information.
- Demo: `https://poloclub.github.io/cnn-explainer/`

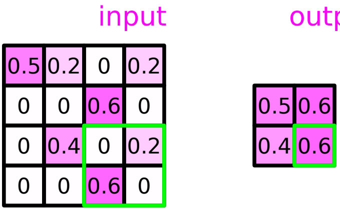
Recap: Convolutional Layer

- Performs a convolution with a **learned kernel**.
- If **multiple input channels**: Learns separate kernels, adds up convolved channels.
- Often produces **multiple output channels** with different sets of kernels.



Recap: (Max) Pooling Layer

- Divides image into **non-overlapping windows**.[1]
- Only **keeps the maximum** value per window.
- Retains **what features** were found, but **not** exactly **where**: shifting the input a little will keep many outputs the same.

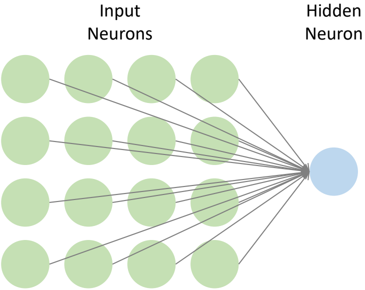


[1] Windows could also overlap, but typically they do not.

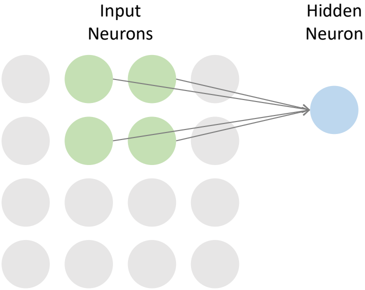
Recap: Fully-Connected Layer

- Convolutional and Pooling Layers are **locally-connected**: Each output depends on a limited part of the input.
- **Fully-Connected** Layers are connected to **each pixel in each input channel** with a separate weight.

→ Useful for producing a **global prediction**, such as a classification.



FNN



CNN

Demonstration

Training a well-working (?) classifier in 5 minutes

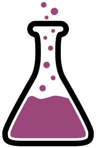
Secret Ingredients

What are the magic ingredients? What does the **model** look like? How can we train it so **quickly** from so **few images**?

Data Augmentation

Dropout

Batch Normalization



Deep Networks

Transfer Learning

Learning Rate Schedules

Data Augmentation

- For each task, some input properties are **relevant** and some are **irrelevant**. Some of them are **easy to modify**.

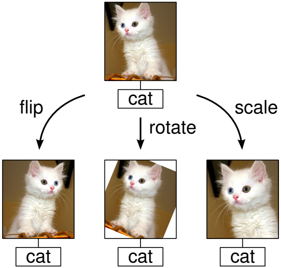
cat, facing left

cat, facing right

- The model will use **all properties** that help predict targets on the **training data**.
- With **careful data modifications**, we can help the model to learn **what we know**.

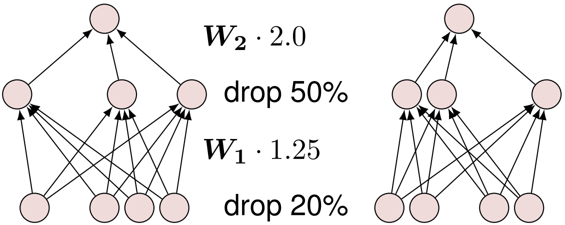
Data Augmentation

- We modify training examples by
  - changing an **irrelevant** input property and keeping the target as is, or
  - changing a **relevant** property and computing the new target.



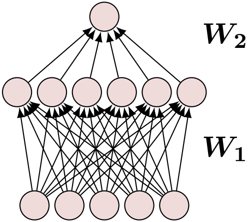
- This **encourages** the model to use/ignore particular properties.

Dropout



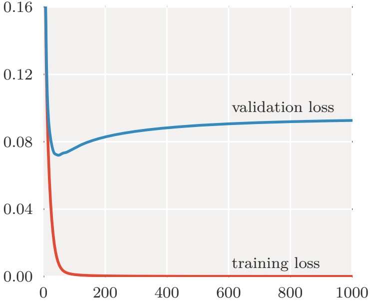
- For each training example, **randomly omit $p\%$ of the units** and scale the remaining weights with $\frac{1}{1-p}$ to compensate.
  - Units cannot rely on all neighbors: Each unit must become useful on its own.
  - Units cannot rely on all predecessors: Each unit must connect to a group of units.

Dropout



- At test time, use the full network.
  - Can be seen as training 2^N networks, then averaging over them at test time.

Dropout



no dropout

dropout

- Strong generic regularizer.
- Typically higher training error, more noisy curve, but reduces overfitting.
- Hinton et al. (2012): Improving neural networks by preventing co-adaption of feature detectors

Batch Normalization

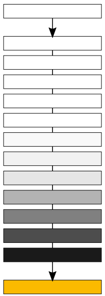
- Models benefit from standardized input data.
- Easy for the input data:
  - Compute mean and standard deviation once over the training set.
  - For each training or test example, subtract precomputed mean, divide by precomputed standard deviation.
- Can we also standardize data for the hidden layers?
  - Cannot precompute once, data distribution changes with each weight update.
  - Cannot afford a full pass over the training set after each weight update.

Batch Normalization

- Ioffe and Szegedy (2015): Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
- Santurkar et al. (2018): How Does Batch Normalization Help Optimization?
- Normalize each mini-batch of training examples by its mean and standard deviation, after every convolution or fully-connected layer.
- Leads to noisy estimate, but noise helps against overfitting.
- At test time, use statistics from training set.

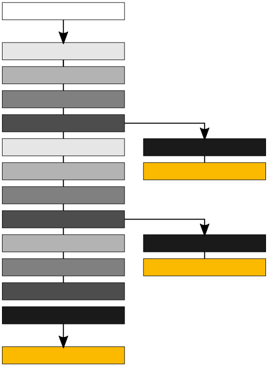
Deep Networks

- Adding **more layers** enables modeling **more complex** functions (more effectively than enlarging existing layers, see Hastad et al. (1986), Bengio and Delalleau (2011)).
- But: Naively stacking many layers can **impede training**!
- **Vanishing Gradient Problem:** Error signal from output layer does not reach the earliest layers.



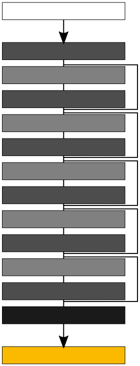
Deep Networks

- **Auxiliary Classifiers:** Additional output heads trained with the same target,
  Szegedy et al. (2014): Going Deeper with Convolutions



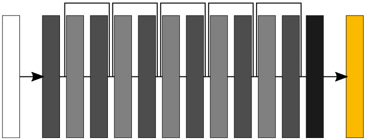
Deep Networks

- **Residual Connections:** Adding a layer's input to its output creates a gradient shortcut,
  He et al. (2015): Deep Residual Learning for Image Recognition

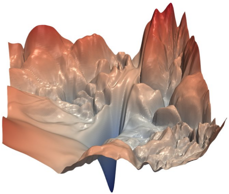


Transfer Learning

- Looking at an image classifier trained on 1000 classes, the **features** detected by the convolutional layers are **pretty generic**: `https://distill.pub/2017/feature-visualization/appendix/`
- We can take such a model, **remove** the last layer, **add** a new layer, and train it on our data.
- Most software frameworks maintain a “Model Zoo” of pretrained models.



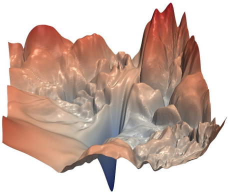
Learning Rate Schedules



Li et al. (2017): Visualizing the Loss Landscape of Neural Nets

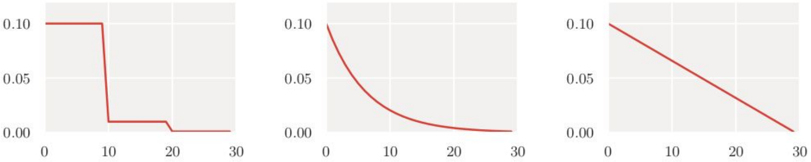
- The learning rate controls the size of steps in the loss landscape.
- Closing in on a local minimum may require careful steps (a low learning rate).
- Getting in the vicinity of a minimum may require a large learning rate.
  - Progress could simply be too slow otherwise.
  - Could get stuck in a worse local minimum.

Learning Rate Schedules



Li et al. (2017): Visualizing the Loss Landscape of Neural Nets

- Exploration vs. exploitation: start large, then decay.
- Common schemes: steps, exponential, linear:

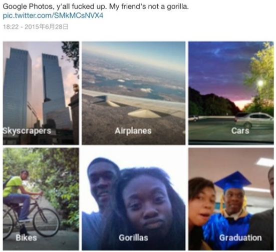


What's the Catch?

- If this is **working so well**, do we still need humans?
- **Beware**: "working well" vs. "reproduces labels on test set"

→ Algorithmic bias

- What if the test set is incomplete?
- What if the labels are unfair?



Jacky Alciné (2015-06-28) via Twitter

What's the Catch?

- If this is **working so well**, do we still need humans?
- **Beware**: "working well" vs. "reproduces labels on test set"

→ Algorithmic bias

→ Adversarial examples

  - CNNs do not process images the same way as humans
  - Can alter images so a CNN changes its prediction, but humans do not
  - Can alter training images so a CNN learns wrong classification boundaries

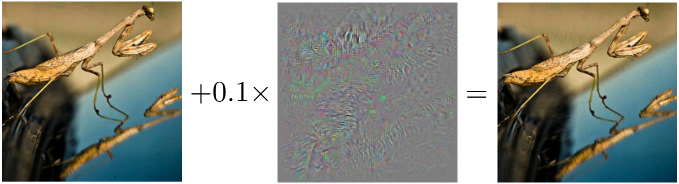
- There is no pretrained model available for every domain and task (and if there is, it might be biased...)

What's the Catch?

- If this is **working so well**, do we still need humans?
- **Beware**: "working well" vs. "reproduces labels on test set"

$\rightarrow$ Algorithmic bias

$\rightarrow$ Adversarial examples



mantis — perturbation — ostrich

Szegedy et al. (2013): Intriguing properties of neural networks