DLVC – Questions Exam 2022S

What is the task definition of image classification? Explain at least 5 challenges and give examples. What is object detection and how does it differ from classification? Which one is harder and why?

Image Classification:

Definition:

Given a finite set of class labels (e.g. {dog, bird, cat})
→ Which class does a given image belong to

Challenges:

- Pose and Viewpoint
- Illumination
- Deformation
- Occlusion
- Background
- Intraclass Variation

Object detection:

Definition:

Given a finite set of class labels (e.g. {dog, bird, cat})
→ Draw bounding boxes around all class instances
→Assign the correct class label to each bounding box

Object detection is more challenging than image classification because in addition to the challenges present for image classification other challenges arise.
The problem is harder to implement efficiently and generally more complex.
There is no hope for using very simple models which may be okay in image classification.

Assume a company asks you to develop an application that is able to predict which kind of bird is depicted in a given image. Which kind of task is this? List and explain the individual steps you'd follow to solve this problem using deep learning.

This task is an instance of **Image Classification**.

I would follow the following steps:

1. Acquire a suitably large, labeled training set of the relevant kinds of birds
2. Use transfer learning to re-use existing models' capabilities with regards to general feature extraction. This will significantly cut down on training time and the required training set size.
3. Optimally select a pre-trained model which was trained on a related domain (birds, nature, small animals, etc.). The closer the better
4. Cut off the last layers of the pre-trained network and replace them with fully connected layers. These layers are the only ones which can be changed during training.
5. If this does not work I would try to remove the training restriction on the feature extraction layers to allow fine tuning of the whole model.

In general, the following "ingredients" are required to do deep learning.

1. A suitable dataset
2. A suitable loss function (e.g. Cross-Entropy)
3. An algorithm for computing the gradient of the loss function (e.g. Backpropagation)
4. An algorithm for updating the weights of the network (e.g. ADAM)
5. A metric for estimating the model's performance (e.g. Accuracy)

What is the motivation for solving vision tasks via machine learning? What is a machine learning algorithm? How are such algorithms used for solving vision problems? What is deep learning in this context and what are its benefits?

Motivation:

Vision tasks are really complex problems.
Crafting algorithms which can directly classify and image or detect objects is almost impossible because of the sheer complexity involved and because the author of the algorithm might not know which properties of the input are event relevant to the task.

Machine learning can learn arbitrary decision boundaries and can therefore be used to solve these very hard problems

Definition Machine Learning:

Algorithms that are able to learn from data and whose performance improves over time given enough training.

Definition Deep Learning:

Deep learning uses neural networks with hidden layers that are neither directly connected to the output nor the input.

Their benefit is that they may be able to learn complex features of the input image which can be used to solve the vision problem the network is optimized to address.

What are class scores? What is the softmax function and why is it useful in this context? What effect does softmax have on the distribution of the input values?

In takes which require classification, class scores are unit-less values assigned to each of the outputs of the neural network. Each score corresponds to one of the previously defined finite classes the model is tasked with identifying.

Class scores are unbounded and can even be negative. Intuitively they can be hard to interpret, and they do not constitute a valid probability distinction where all elements sum up to one!

The softmax function turns class scores into a valid PMF.
Larger values are emphasized, and small ones suppressed. Also, all of the values end up positive.

$$\text{softmax}_t(\mathbf{w}') = \frac{\exp(w'_t)}{\sum_t \exp(w'_t)}$$

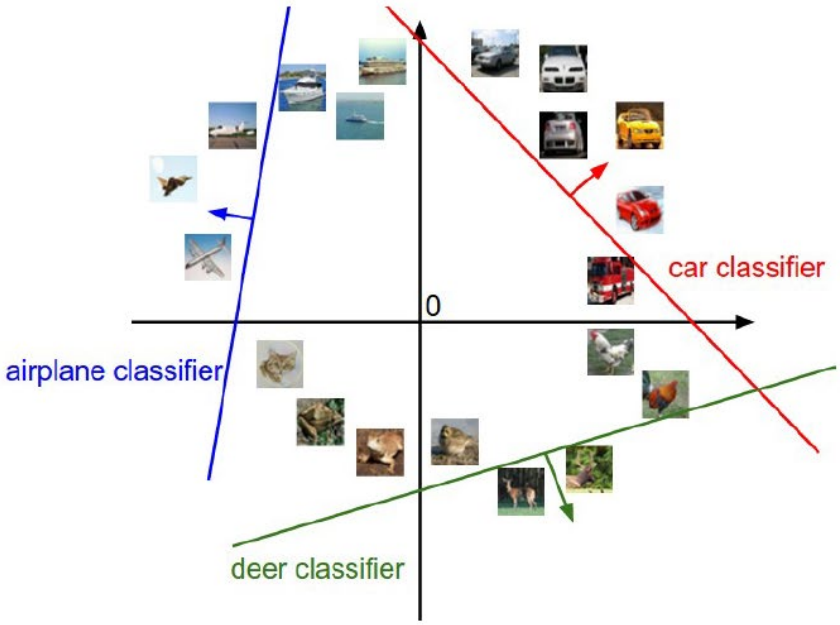
What is a linear model, which types of parameters does it have, and what do they specify? Draw a sketch assuming two-dimensional feature space and three different classes. Draw a few samples per class so that the classes are linearly separable. Draw the decision boundaries a linear classifier might learn and explain how the individual boundaries are related to the classifier output.

Linear models are able separate instances using linear hyperplanes. They are parametric models with two parameters **W,b** which represent a weights matrix and a bias vector.

Dimensions:

W = D*T
b = T

D … Dimensions of the input
T … Number of possible output classes



Linear classifiers are usually used at the end of DNNs. This works because the features extracted before are so good as to enable simple linear classification.

What is the purpose of a loss function? What does the cross-entropy measure? Which criteria must the ground-truth labels and predicted class-scores fulfill to support the cross-entropy loss, and how is this ensured?

Purpose of a Loss Function:

Measures the performance of the function represented by the model.
A lower values is better!
Is calculated for a given dataset and given model parameters

Cross-Entropy

The cross entropy measures the entropy BETWEEN two distributions.

$$H(\mathbf{u}, \mathbf{v}) = -\sum_{t=1}^{T} u_t \ln v_t$$

Using **u** to encode the ground truth label and **v** as the output of the softmax layer, the cross entropy describes how well the predicted probabilities fit the ground truth.

Cross entropy can only reach 0 if the ground truth is exactly 1 for only one class.

This is ensured by just setting u to 1 at the ground truth index and 0 everywhere else.

What is the purpose of optimization in the context of machine learning? How does the gradient descent algorithm work? What is the gradient of a function? Explain the terms learning rate and step size, and how they are related.

Purpose of optimization:

Use some algorithm to change the model parameters in a way that minimizes the loss function.

This leads to better train set performance and hopefully a better model overall.

Note that the loss function is not linear with regards to the model parameters, therefore a non-linear optimization algorithm is desirable.

Gradient Descent is a popular choice for this.

Gradient Descent:

Gradient Descent is an iterative optimization algorithm which changes the model parameters according to their respective gradients of the loss function.

The gradient of a function with regards to some parameters is a vector which points into the direction that these parameters need to be changed to maximally increase the function.

Gradient descent uses the opposite direction to update the parameters.

Step size:

The defines how far the parameters get updated with regards to the gradient magnitude and the learning rate.

In every iteration we

- Compute gradient $\boldsymbol{\theta}' = \nabla L(\boldsymbol{\theta})$
- Update parameters $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\boldsymbol{\theta}'$

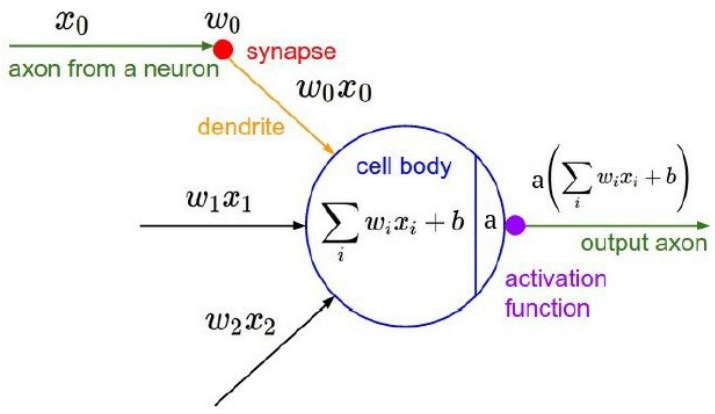
Hyperparameter $\alpha > 0$ is called learning rate

- Final step size is $\alpha \|\boldsymbol{\theta}'\|$

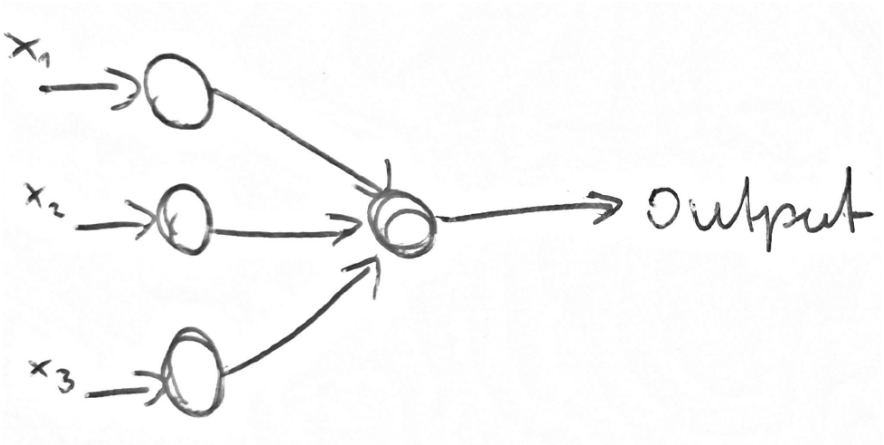
What is the definition of a neural network? Draw a multi-layer perceptron for binary classification of three-dimensional feature vectors that has a single layer with four neurons, using circles to represent neurons and arrows for data flow. Which operation does a single hidden layer neuron perform?

Definition:

A neural network is a collection of artificial neurons.



Drawing:

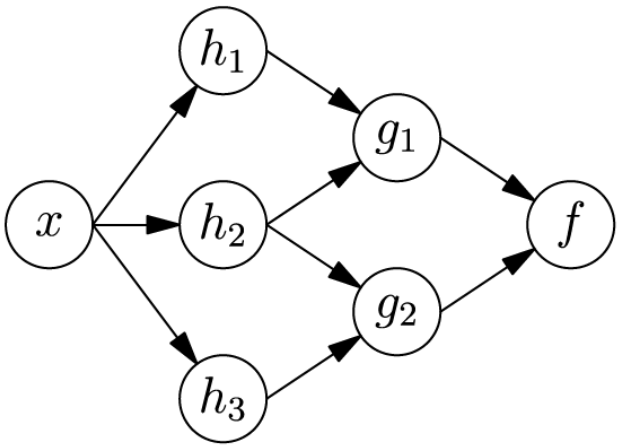


Operation of a single neuron:

Simple linear function:

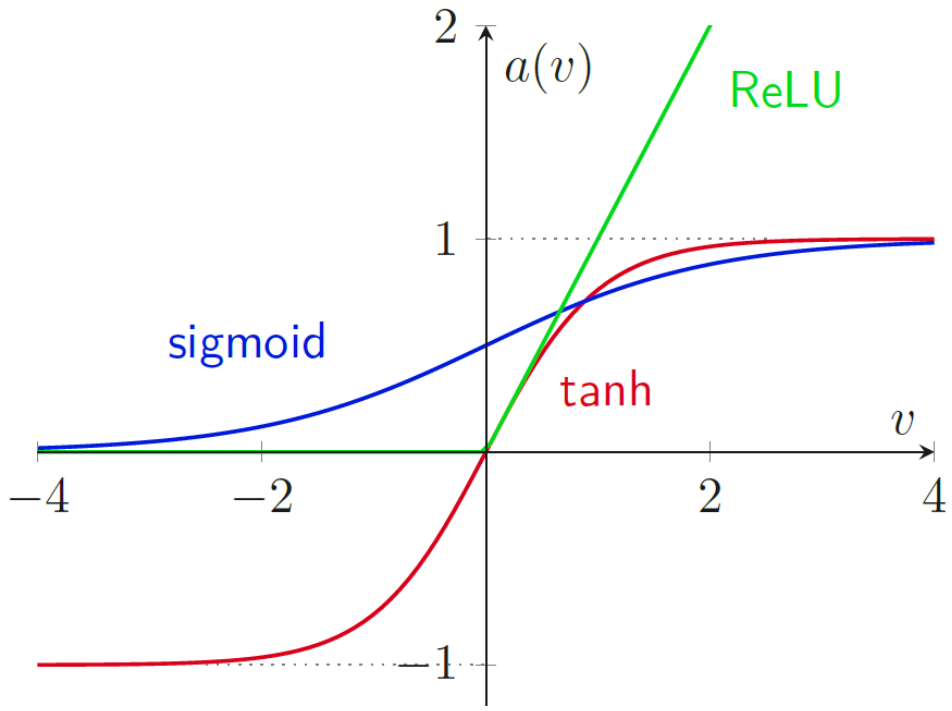
$$o = a(\mathbf{w} \cdot \mathbf{x} + b)$$

$$o = f(x) = f(g_1(h_1(x), h_2(x)), g_2(h_2(x), h_3(x)))$$



What is an activation function? Draw graphs of at least three activation functions that were covered in the lecture, including ReLU. Which activation function is the most popular in deep learning and why?

Function used to decide the value which is put out by a neuron depending on the inputs + the neuron bias.



ReLU is the most popular loss function.

An advantage of ReLU is gradient calculation is simpler due to the high frequency of neurons which put out 0.

Also the calculation is way cheaper compared to other loss functions.

What is the purpose of the backpropagation algorithm, how does it differ from the "naive" algorithm for the same purpose, and what are its advantages? Explain the steps of the algorithm at a node of a computational graph. What are the gradient flow patterns during the backward pass through x+y, x · y, and max(x, y) nodes?

Naïve approach:

Neural networks can be treated as computational graphs, i.e. a function mapping inputs through a network of other functions.

Recursive applications of the chain rule can be used to get a derivative for such computational graphs.

Recall that if $F(x) = f(g(x))$ then $F'(x) = f'(g(x))g'(x)$

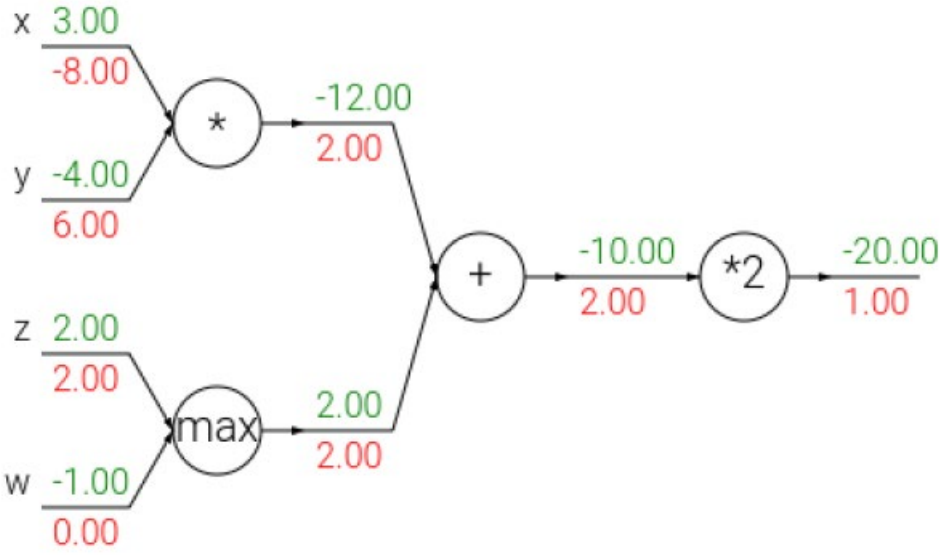
The naïve approach would be to calculate the local gradients for each node independently.
But this is pretty bad for large networks.

Backpropagation:

Only perform one calculation for the output node with regards to all nodes before it and store intermediate results for efficiency.

Gradient flow patterns:

x+y: Distributed equally to input nodes
x*y: Routes gradients by multiplying with the switched inputs
max(x,y): Route gradient to the largest input



What is a feature in the context of machine learning? How do low-level features differ from high-level features? How can high-level features be obtained from images? What are the steps of the traditional image classification pipeline?

Features in ML:

Features are the inputs that an ML model receives. The output of the model depends on these features.

Low-Level features are things like border, color, contrast and other "directly calculable" properties of the input.

High-Level features tell us more profound things about our input and can be very task specific. (Outside or inside, how pointy are the ears, fur color, season of nature outside)

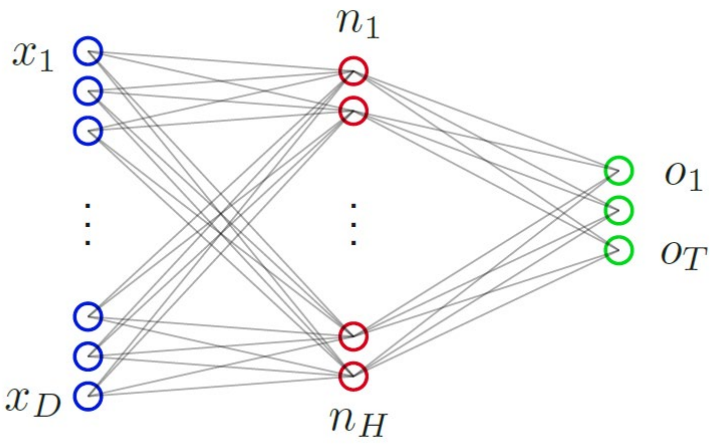
High level features can be obtained by using convolutional layers for images and can result in very relevant and information dense feature vectors.

Traditional image classification pipeline:

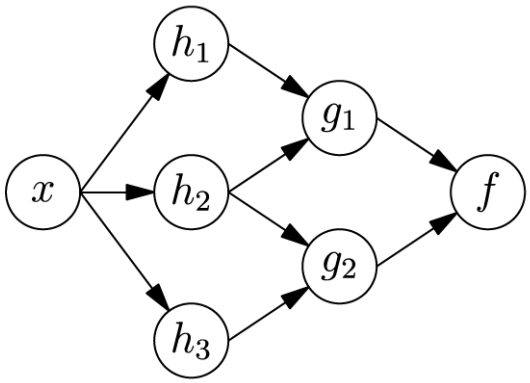
1. Extract low-level feature vectors
2. Optionally perform dimensionality reduction (e.g. PCA)
3. Process using some machine learning classifier.

What is an MLP? Draw a sketch of one for binary classification of three-dimensional feature vectors and one hidden layer with five neurons, and explain the sketch. Why do MLPs struggle with vision tasks?

MLPs are neural networks with hidden layers that are not connected directly to the output or the input, but only to other neurons.



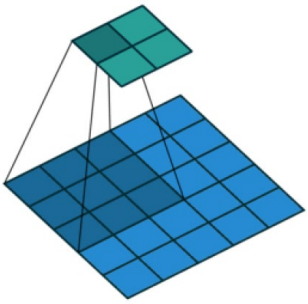
Sketch for MLP with 5 neurons:



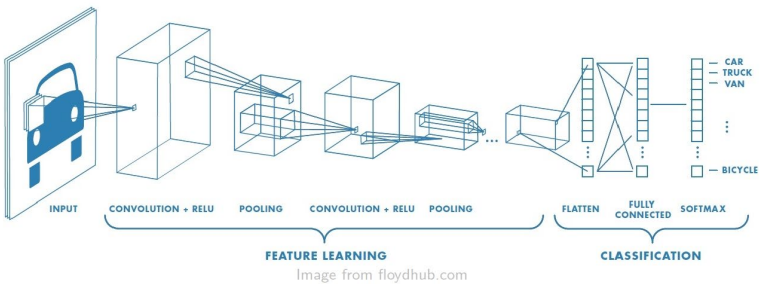
MLPs struggle with vision tasks because they cannot extract any high-level features. The neurons only depend on a very small neighborhood of neurons.

What is a convolutional neural network? Create a sketch that illustrates with layers such a network usually has, by example of image classification. Briefly explain the purpose of each layer type in the sketch.

CNNs use convolutional operations to change the dimensionality of layers.



This enables the extraction of features at different levels and enables the network to be more flexible and capable.

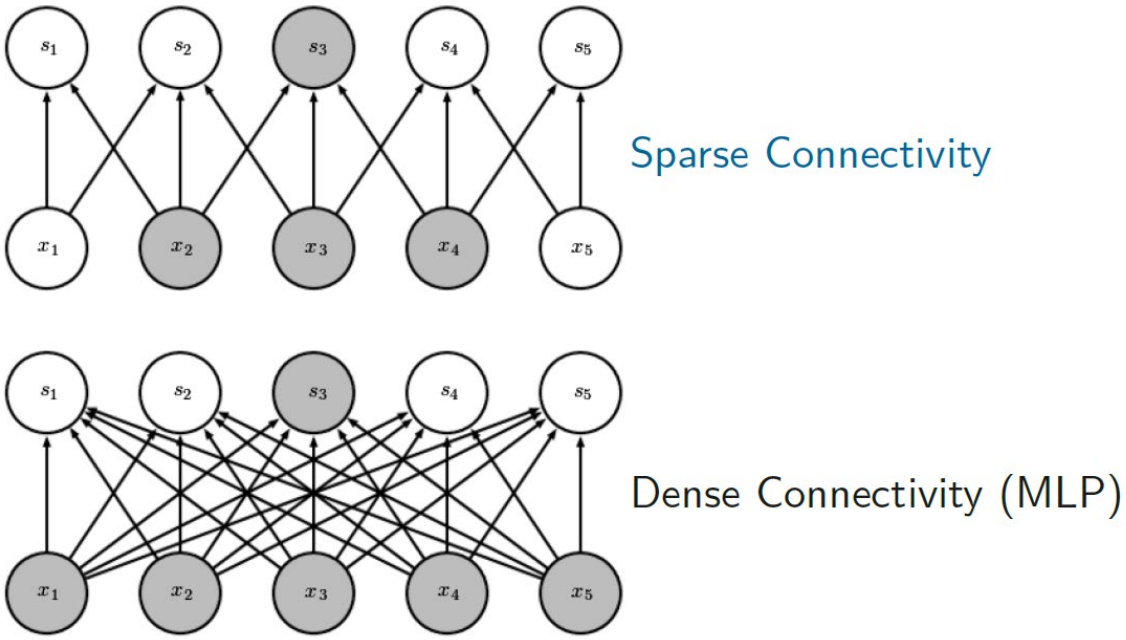


Explain the differences between dense and sparse connectivity, and draw a sketch for illustration. Why and where is the concept of sparse connectivity used most prominently in neural networks for vision applications? What is the receptive field of a neuron and how does this concept relate to sparse connectivity?

Sparse VS Dense Connectivity:

Sparsely connected networks feature connections to only a few neurons in the next layer for a given neuron.

In densely connected networks each neuron of a given layer is connected to every neuron in the next layer.



Sparse connections are used for feature extraction layer so that local features can be calculated for spatially close neurons.

The receptive field of a given neuron are all the input neurons that information is received from.

What is a convolutional layer? What are feature maps? Why are parameters shared within a feature map? What operation does such a layer compute, assuming an input tensor with shape 3×32×32, a 3×3 kernel, and 16 feature maps? What is the shape of the output tensor?

Convolutional layers are used to change the dimensionality of a tensor by using a fixed kernel which is moved over the input.

Feature maps are extra repetitions of the convolutions which allow for different features to be extracted. This corresponds to repeating a convolutional operations multiple times to learn different features. Each feature map learns its own kernel and can therefore extract different features.

Input: 3x32x32
Kernel: 3x3
Feature Maps: 16

Result:

16x10x10

This assumes a stride length of 1 and no padding!

What is representation learning? Explain the overall approach of representation learning using convolutional neural nets. What is deep learning?

It's very hard to extract meaningful features from inputs which can then be used to predict properties.
Using feature extractors is a good idea, but hand-crafting them is infeasible.
Representation learning refers to the process of using a neural network to extract meaningful features.
Deep learning is representation learning with DNNs.

When using many layers of neurons an proper architecture low-level features can first be obtained which lead to the extraction of meaningful high-level features.

What is pooling? Explain at least two layer types for this purpose. Also explain at least two layer types for the opposite purpose. What is global average pooling and where is it used?

Convolutional layers roughly retain the input size (depending on the padding), but this slows down computation.

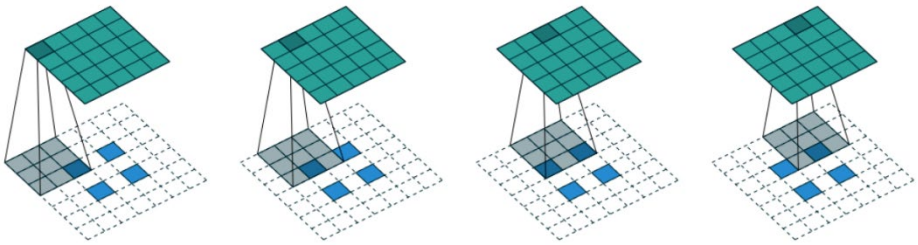
Therefore, we want to reduce the size of the tensors using local aggregation. The idea is that because of strong local relationship, this doesn't loose to much information.

Types of pooling layers:

- Max Pooling
- Mean pooling
- Strided convolutions.

Types of __ layers:

- Upsamling
- Transposed convolutions



Stride length 1/s

- Subpixel convolution
  Works by shuffling together different channels and then doing a simple stride 1 convolution operation.

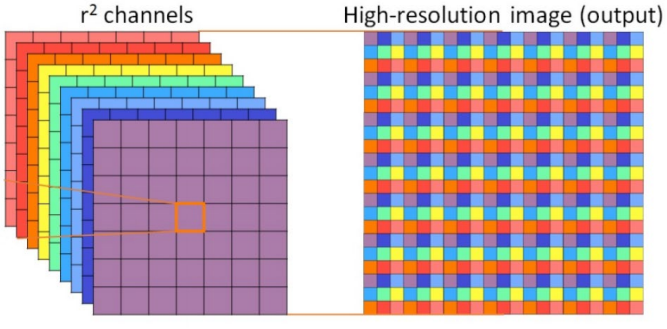


Image from [4]

What are the three overall steps that comprise one iteration during training? What is the gradient descent update rule? What is minibatch gradient descent and why is it used in practice? What is an epoch?

Steps for one training iteration:

1. Calculate the loss function L
2. Compute the gradients for each node using back propagation
3. Use gradient descent to update the model weights

Gradient descent update rule:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla L(\boldsymbol{\theta})$$

Change the weights according to the learning rate and the gradient magnitude (in the negative direction).

Usually stop if the change in the loss function reaches a pre-defined small value.

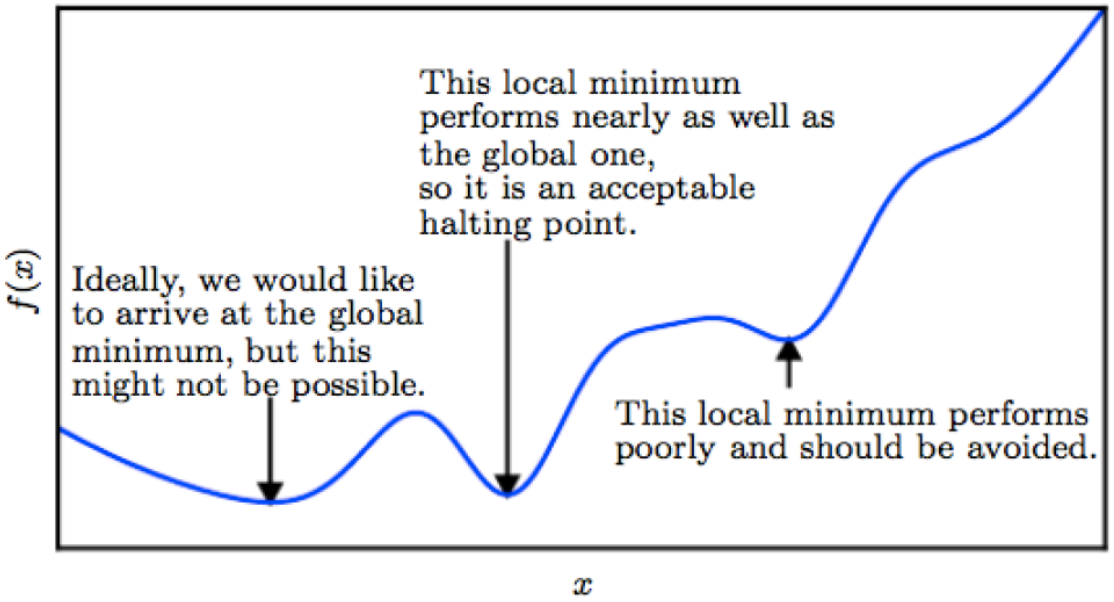
Minibatch gradient descent:

Pass a fixed number of samples at once and only backpropagate afterwards. This saves time and is still effective.

Epoch:

An epoch is one full run through the training set. Usually, multiple epochs are used in training.

What is the difference between a local and a global optimum? Draw a sketch that illustrates the difference. Are local minima a problem in deep learning? Why (not)? What is momentum and why is it beneficial?



Local minima can be a problem if they are too far away from a global minimum. Momentum is a term which defines "velocity" at which the parameter updates are moving.

This term can help to escape local minima by temporarily accepting worse solutions.

Iteration of gradient descent with momentum

- Update velocity $\mathbf{v} = \beta\mathbf{v} - \alpha\nabla L(\boldsymbol{\theta})$
- Update parameters $\boldsymbol{\theta} = \boldsymbol{\theta} + \mathbf{v}$

In this formula, Beta is the momentum parameter.

What is the Adam optimizer, and what are its advantages over gradient descent? Explain in your own words and highlight the differences. Formulas are not required.

Adam combines RMSProp with the notion of momentum.

RMSProp:

Basic idea is to reduce oscillations by adapting high variance parameters less. This means that the choice of Alpha is not as critical

Update step (initially $\mathbf{n} = \mathbf{0}$)

- $\mathbf{n} = \beta_2 \mathbf{n} + (1 - \beta_2)\nabla^2 L(\boldsymbol{\theta})$
- $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla L(\boldsymbol{\theta}) / \sqrt{(\mathbf{n} + \epsilon)}$

ADAM:

Update step (initially $\mathbf{m} = \mathbf{n} = \mathbf{0}$)

- $\mathbf{m} = \beta_1 \mathbf{m} + (1 - \beta_1)\nabla L(\boldsymbol{\theta})$
- $\mathbf{n} = \beta_2 \mathbf{n} + (1 - \beta_2)\nabla^2 L(\boldsymbol{\theta})$
- $\mathbf{m} = \mathbf{m} / (1 - \beta_1^t)$
- $\mathbf{n} = \mathbf{n} / (1 - \beta_2^t)$
- $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{m} / (\sqrt{\mathbf{n}} + \epsilon)$

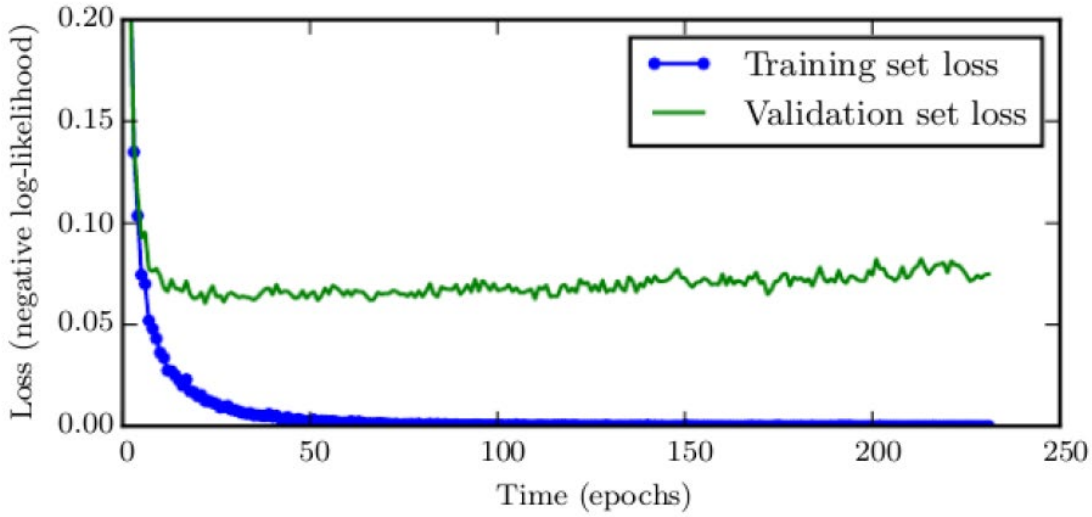
ADAM can adapt the learning rate by using a mix of RMSProp and momentum. This makes it a good choice as a general-purpose learning algorithm.

Explain the terms overfitting and underfitting. How can these issues be spotted during training? Create graphs for illustration.

Overfitting:

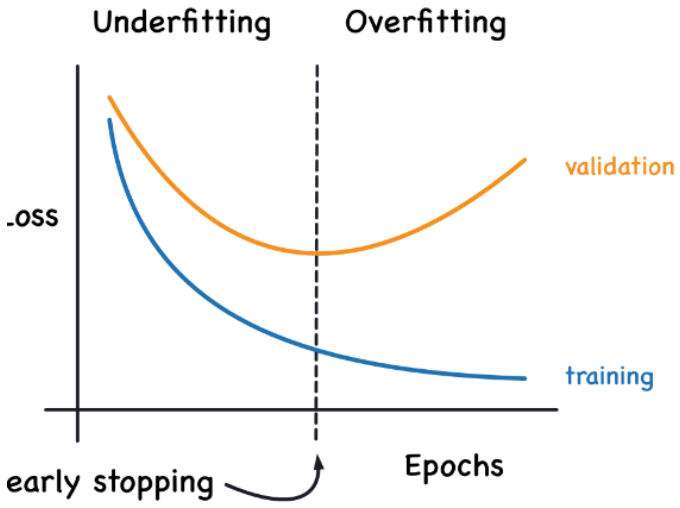
Very high variance fit on the training data which hurts generalizability of the model.

Can be spotted if training loss goes down but validation loss stays high.



Underfitting:

Data is not well modeled at all!
Validation loss and training loss are comparable, but bad.



Explain the purpose of batch normalization as well as the operations a batch normalization layer performs (math is allowed but not required). What ensures that networks with such layers can be used with single inputs after training?

Batch normalization aims to estimate the variance and means of the current batch and uses this to scale the data currently being passed through the network. The goal is to avoid exploding and vanishing gradient problems.

These layers learn parameters to perform the normalization.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

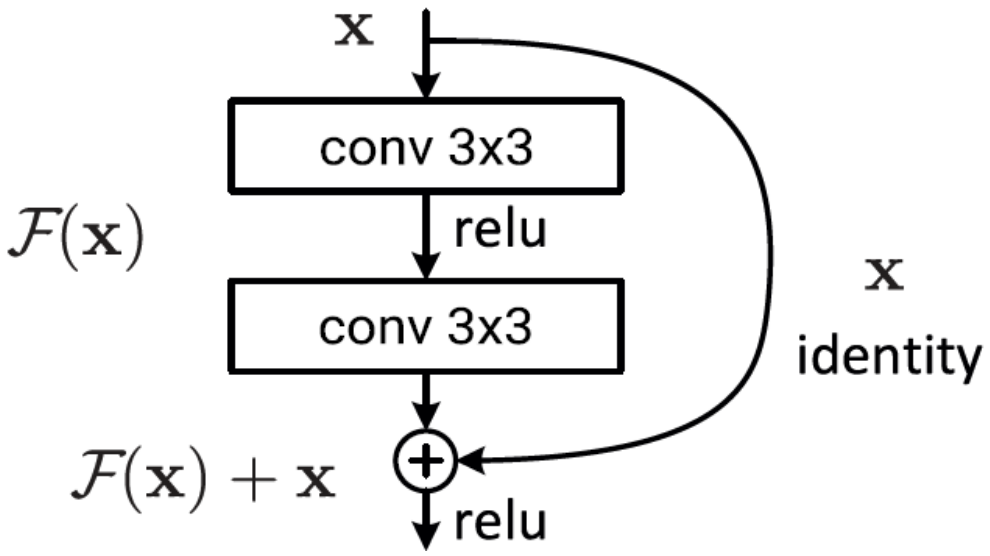
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

If the model sees that the current batch has a mean and variance that is very close to the learned parameters, the layer can be skipped by using the identity function.

To ensure the network can still be used with single inputs during inference, averages for the mini-batch statistics are stored in the model and simply used during the inference phase.

What can very deep networks suffer from, despite using batch normalization? What is a residual network and why are they so popular? Draw a sketch of a residual block and explain the signal flow.

Because if there are layers which are unnecessary, it would be best for them to just model the identity function. But this is kind of hard to do, so to make it easier for the model skip connections are used.



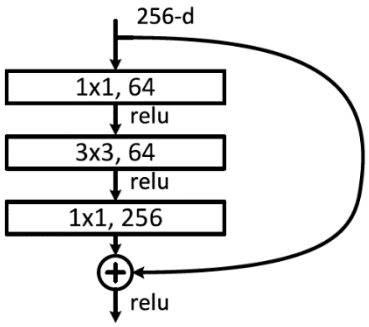
As long as the network just learns small weights for the layers that are skipped (e.g. by using weight decay) the skip connections will make the network behave as if the layers don't really exist.

What is a pointwise convolution and what are its uses? What are bottleneck layers and why are they useful?

Pointwise convolution:

A pointwise convolution has the height and width of 1 and is as many channels deep as there exist.
Basically, reduces the number of channels, which can be very useful before expensive operations.
If the stride is adjusted H & W can also be changed.

Bottleneck layers:



Bottleneck layers reduce the number of channels before performing some operation. In the case above the operation is a 3x3 convolution.
So the number of channels is reduced from 256 to 64 by using a point-wise convolution.
After the operation the dimensionality is increased again to enable the skip connection summation.

What is training data augmentation and why is it useful? What would suitable forms of data augmentation be, assuming the task of digit image classification? What is mixup and how does mixup affect ground-truth class scores?

Data augmentation works by applying different transformations (that don't affect the input label) on the training data to:

- Increase the data set size
- Introduce more variety to make the model more robust

This can be performed online, so the size of the stored data set is not affected.

For images some suitable transformations are:

- Cropping
- Horizontal mirroring
- Similarity or affine transformations
- Contrast, Brightness, Sharpness changes

Mixup works by blending two input images along with their class labels. The ground truth vectors get blended as well!

What is regularization? How does weight decay change the loss function, and how does this affect the gradient descent update rule? What is dropout?

Regularization aims to prevent very large weight parameters by penalizing them. This should reduce the validation/test performance and reduce overfitting.

However, training performance could be diminished (which is okay as long as test/validation is good).

The loss is changed by adding a term that depends on the weight decay parameter and the squared absolute sum of all weights in the network. This means the model is incentivized to reduce the weights.

$$L_{reg}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \frac{\delta}{2}\|\mathbf{w}\|^2$$

$\mathbf{w} \subset \boldsymbol{\theta}$ is vector of all weights

$\delta \in (0, 1)$ controls amount of regularization

- Usually $\delta \in [0.0001, 0.01]$

Gradient descent turns into:

$$\begin{aligned} \mathbf{w} &= \mathbf{w} - \alpha(\nabla L(\mathbf{w}) + \delta\mathbf{w}) \\ &= \mathbf{w} - \alpha\delta\mathbf{w} - \alpha\nabla L(\mathbf{w}) \\ &= (1 - \alpha\delta)\mathbf{w} - \alpha\nabla L(\mathbf{w}) \end{aligned}$$

When tackling a new problem via deep learning, what are the questions you should ask yourself? Answer these questions for the problem of detecting dogs in images.

Recall our ingredients for image classification

- A suitable dataset
- A network with a suitable final layer (linear layer)
- A suitable loss function $L(\boldsymbol{\theta})$ (cross-entropy)
- An algorithm for computing $\nabla L(\boldsymbol{\theta})$ (backpropagation)
- An algorithm for updating $\boldsymbol{\theta}$ on this basis (Adam)
- An intuitive performance metric (accuracy)

So when tackling a new problem, ask yourself

- Do I have enough (labeled) data?
- What is a suitable network output?
- What is a suitable loss function?
- What is a good metric for evaluation?

For dog detection:

Do I have enough (labeled) data?:

This should present no problem. There exist pre-labeled datasets in the animal domain

What is a suitable network output?:

The positions of the top left corner of the box containing dogs along with their height and width.

What is a suitable loss function?:

Localization loss → How well do the predicted boxes overlap with the real positions of dogs in the image

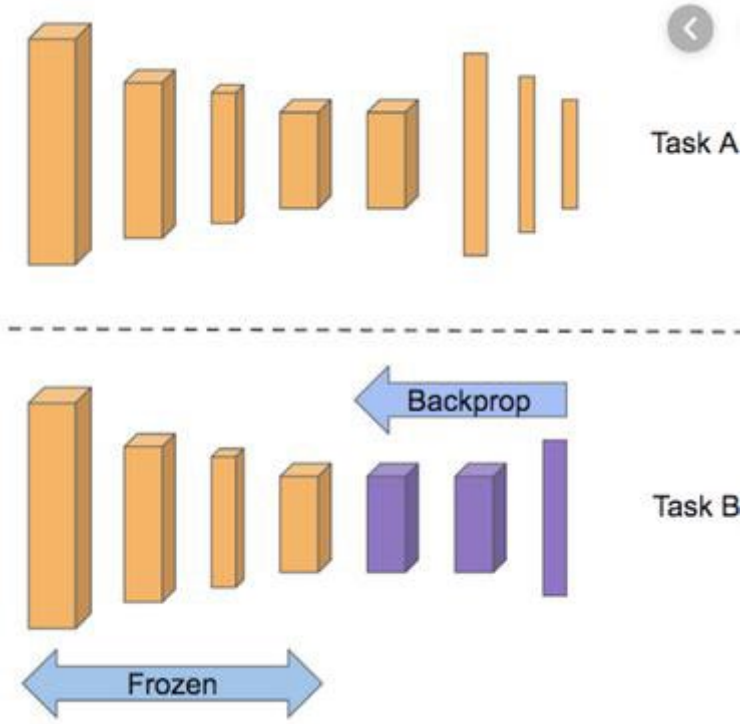
What is a good metric for evaluation?:

Average Precision (AP)

Explain the terms transfer learning, pre-training, and fine-tuning. Why is transfer learning so important in deep learning? Explain at least two tasks that benefit from transfer learning, and how it would be applied.

Many of the tasks performed in a network (especially low-level feature extraction) are not domain specific. Additionally, learning these model functions often take a very long time and require a lot of resources.

Therefore, a model which is already trained for a different problem can often be reused by taking the first few layers of the network.



Depending on the similarity of the tasks fewer or more layers might need to be replaced.

Transfer learning is important because Deep Learning requires very large data sets which are just not available for many tasks. Also, the training time often prohibitive.

Pre-Training

Training the network on a different task (often performed by someone else)

Fine-Tuning

Initially only the newly added layer will be trained, but it might be required to unfreeze the earlier layers later the re-train them a bit as well. Especially when the required high-level features are different. This step of retraining the entire network is called fine-tuning.

Tasks that benefit from transfer learning:

- Medical applications where gathering large datasets is infeasible. E.g. predicting cancer in chest mammography scans.
- Very specific applications like identifying manufactured parts of a single company for inventory purposes.

Explain the concept of object detection using deep learning and region proposals.

The idea of region proposals is to first generate a large pool of possible regions and classifying and ranking these regions.

This solves any problems which arise when using fixed size or quantized window sizes.

However, the naïve approach of using region proposals on the high-resolution input directly is pretty slow. A partial solution to this issue is to perform the region proposal step on a suitably sized conv layer output, but it's still not optimal.

What is the conceptual difference between R-CNN and Fast R-CNN, and between one-stage and two-stage detectors?

R-CNN VS Fast R-CNN:

The difference between R-CNN and Fast R-CNN is where to create the regions used for region proposal.

Fast R-CNN does this after some convolutional layers, while vanilla R-CNN performs this step directly on the input image (and is slower because of it)

One-stage VS two-stage detectors:

Two-stage detectors first generate r region proposals and then evaluates them individually.
The time this takes depends on the number of region proposals and can be very slow. This means that two-stage detectors are not suitable for real-time applications but only retrospective analysis.

One-stage detectors perform object detection in a single pipeline. The inference time is independent of the number of regions.

This is done by using fixed anchor boxes with variable size and ratios. YOLO is a popular example of this.

Explain the approach of YOLO. What are anchor boxes? Given a 3 × 3 grid, k anchor boxes, and 5 kinds of relevant objects to detect, what is the shape of the output tensor and what does this tensor encode?

YOLO is a popular anchor-based detector family. Usually, a pre-trained image net variant is used as the backbone of these models.

The backbone pools the image down to the desired grid size!

Anchor boxes are created by first dividing the image into pre-defined grid cells and then overlaying a pre-defined selection of bounding boxes onto these grid cells.
For each of these boxes the following is predicted:

- The class scores
- The offset and scale of the box
- The IoU overlap with the most relevant object.

Calculation:

3x3 grid, k anchor boxes, 5 classes

Tensor = 3x3x(5+C)xk → 3x3x10xk

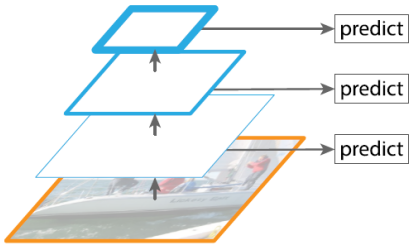
For each grid cell, all k anchor boxes are predicted using 5 features + the class scores.

What is a feature pyramid network, and the motivation for using them? How do they fit into the overall network stages for object detection?

An issue with object detection using anchor boxes is that small object might not be properly detected.

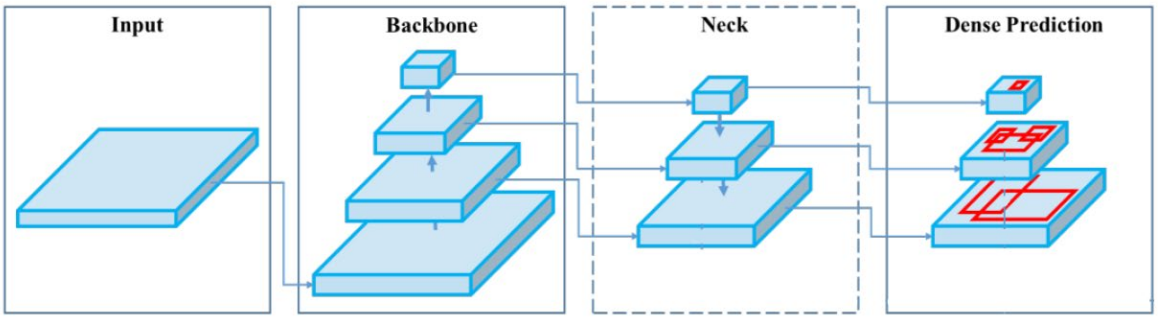
Late is the network the width and height of the tensors are small, so while the features they represent are very informative, their spatial resolution is limited.

This could be addressed by attaching multiple detector heads to varying conv layers.



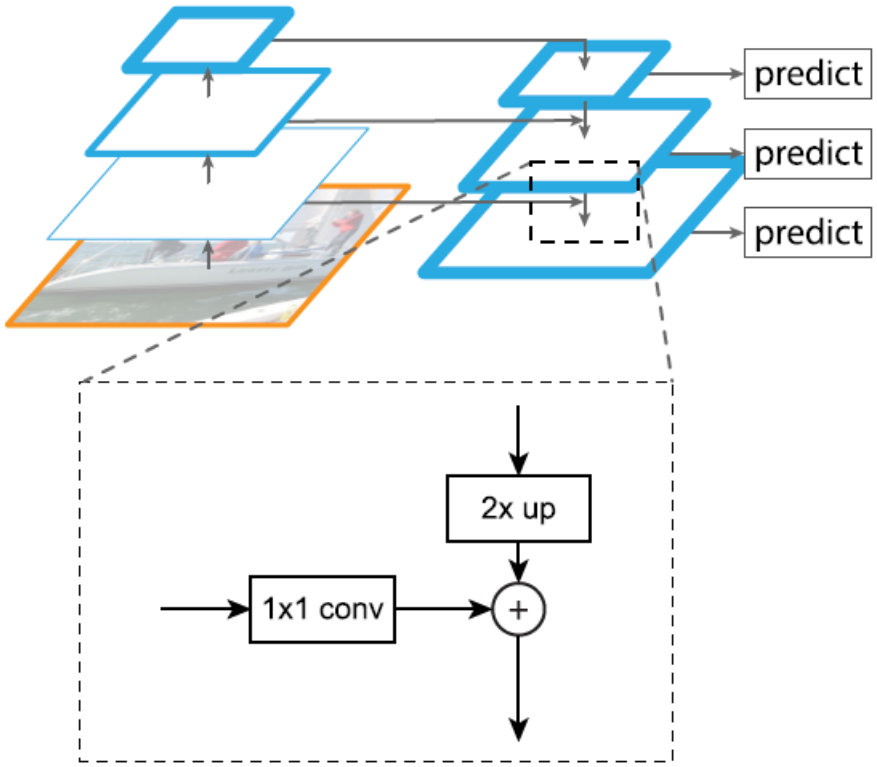
(c) Pyramidal feature hierarchy

Example for YOLO:



The predictions from different pyramid layers are then combined.

Notice that in the neck, lower dimensional tensors are passed on and combined with the higher dimensional tensors using pointwise convolutions.



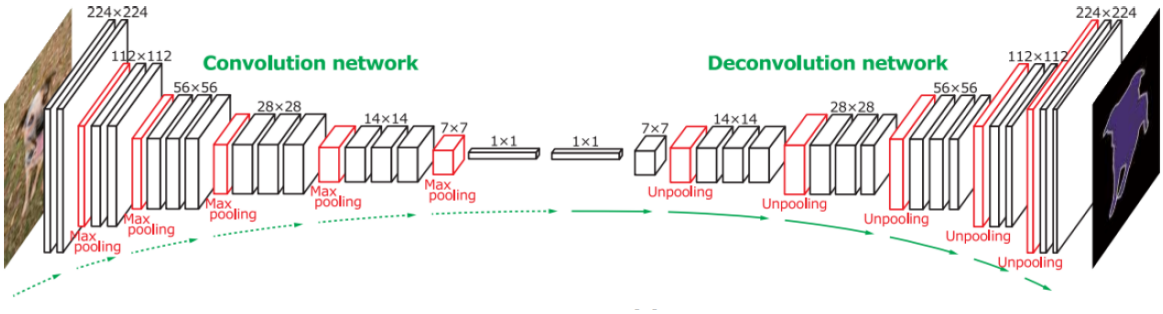
Explain the terms dense prediction and semantic image segmentation. Explain the two overall stages of convolutional neural networks for dense prediction, and draw an illustration.

Dense prediction tasks do not output labels for the entire image or regions of the image, but rather for each pixel in the image!
Therefore the output tensor is very information dense (unlike in simple classification).

In semantic image classification, class labels are assigned to each pixel of the image.
If convolutional layers are used, up-sampling has to be performed to get an output tensor of the same dimensions as the input image.

Two stages of dense predictions with convolution:

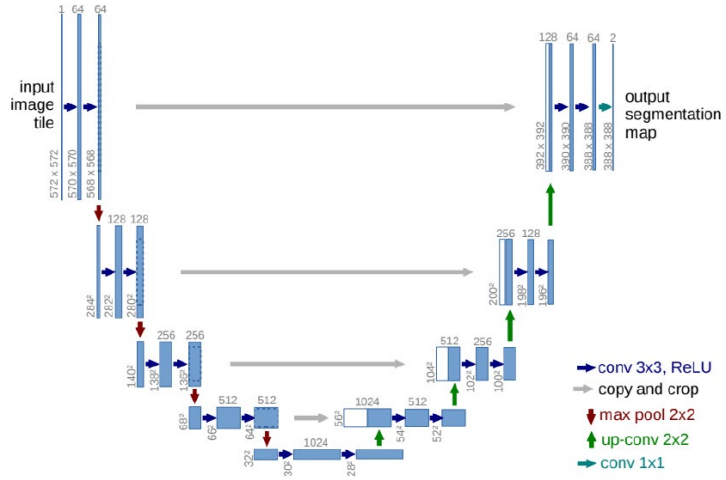
1. Convolution
2. Deconvolution



What is a U-Net, which kinds of skip-connections does it use, and where are these connections placed? Draw a simple chart of such a network and highlight the skip connections. How are signals merged, and why is this not done via summation?

To prevent information loss in down-sampling (because this part is often pre-trained and might remove information which was not needed during pre-training), skip connections are introduced for the up-sampling layers to make sure all information from the original image is available in addition to the high level features learned by pre-trained convolutional layers.

The connections are placed across the two network parts between layers of corresponding dimensionality.



Gray → Skip connections

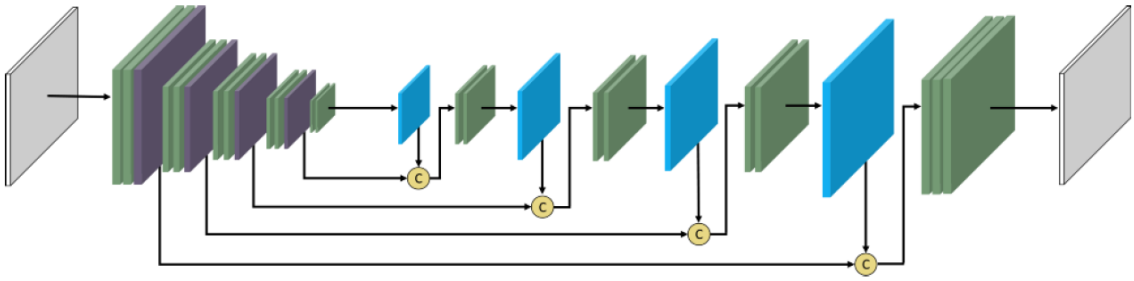
Merging signals:

Signals are merged using concatenation instead of summation.

This is done to preserve more of the information which was potentially removed during the convolution layers. Additionally, back-propagation is simplified somewhat.

Assume you are tasked with creating a neural network that can remove written text and compression artifacts from images. Draw a sketch that shows the overall network architecture, and explain it. How can training data be obtained? What kind of loss is suitable for this task?

A U-Net would be a good option for this task!



Training data could be obtained by algorithmic image augmentation. This would entail manually introducing artefacts to artefact-free images and using the difference of the images as training loss.

Cross-entropy or simply MSE loss could be used. (Between the original and reconstructed image)

Explain the overall concept of generative adversarial networks, including the purposes of the two networks involved, and how they interact. What are latent vectors and what are they used for?

GANs work by having two models, a generator, and a discriminator. The discriminator is trained on binary loss as is the generator.

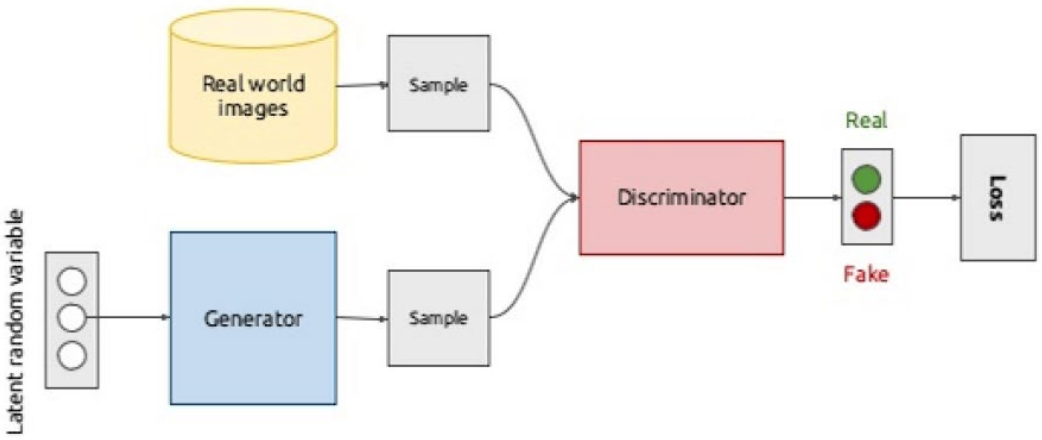
The Generator aims to produce images which seem natural, while the discriminator aims to identify synthetic images as opposed to natural images.

Over the course of the training both networks get better and in the end the Generator can be used to generate realistic images.

The discriminator can be any image classification network (like ResNet)

The latent vectors are a source of randomness which are samples from a pre-defined distribution. This is needed to have variation in the images produced by the generator.

On a high level the generator learns a probability distribution $P(x \mid z)$ and by changing z, other parts of the distribution are sampled.



Training is stopped when the discriminator cannot perform any more meaningful distinctions and outputs (0.5, 0.5) class probabilities for real and fake images.

Explain the training process of generative adversarial networks, including pseudocode of the training loop. What is the ideal outcome?

Training loop:

```
while not done:
  for k iterations:
    sample half of mini-batch from dataset (class real)
    sample half of mini-batch from G (class fake)
    train D on mini-batch (but not G)

  sample mini-batch from G (class real (!))
  train G trough D on mini-batch (but not D itself)
```

First get the discriminator good and then improve the generator.

Then repeat again and improve discriminator.

Training is stopped when the discriminator cannot perform any more meaningful distinctions and outputs (0.5, 0.5) class probabilities for real and fake images.

Possible issues:

- $\mathcal{G}$ cannot fool $\mathcal{D}$ (vanishing gradients)
- $\mathcal{G}$ generates similar outputs regardless of $\mathbf{z}$ (mode collapse)

Pros

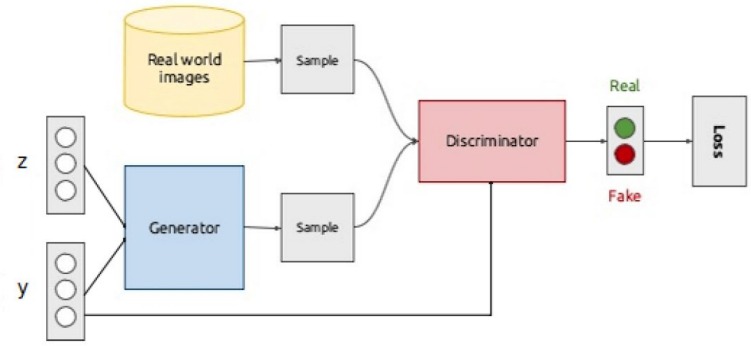
- Among best methods for generating realistic images
- Inference (e.g. image generation) is faster than alternatives
- Flexible framework (next slides)

Cons

- Hard to train (previous slides)
- $\mathcal{G}$ does not actually learn $\Pr(\mathbf{x})$, can only sample from it
- No intuitive way to control what to generate (next slides)

What is a conditional generative adversarial network, and how does it differ from the unconditional variant? Create a sketch that illustrates these differences.

Conditional GANs introduce another input y which can be interpreted to represent the image classes which should be generated.



To generate images from a specific class:

- Choose y accordingly
- Vary z

Can perform many tasks.

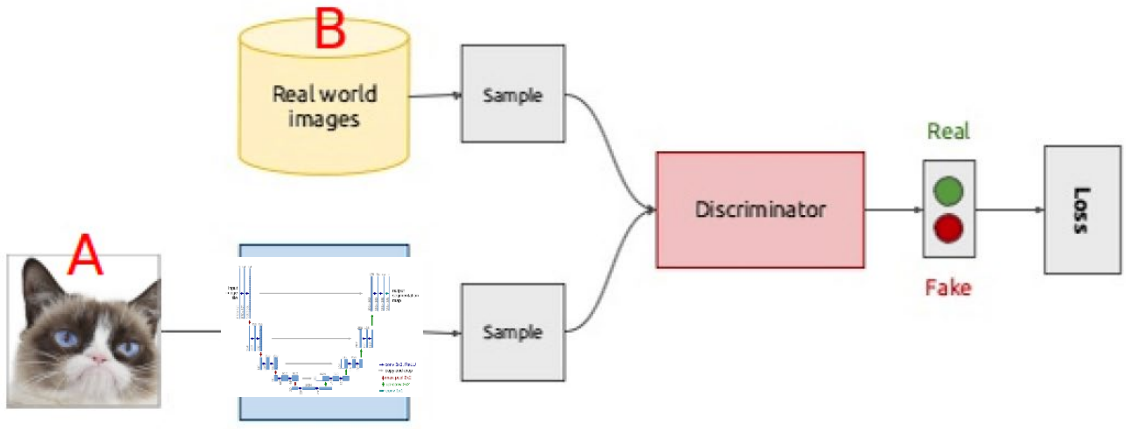
Assume you are tasked to create a generative adversarial network that can colorize grayscale images. Create a sketch that shows the overall architecture. How would the training set look like, and how could one be obtained?

This is a classic Image to Image Translation task

Sketch:

U-Net for generator and some established image classifier (e.g. Res-Net) as discriminator

- $\mathcal{G}$ learns to map images from domain A to domain B
- In sense that $\mathcal{D}$ cannot distinguish between B and $\mathcal{G}(A)$



The training set can be any color image set (e.g. ImageNet).

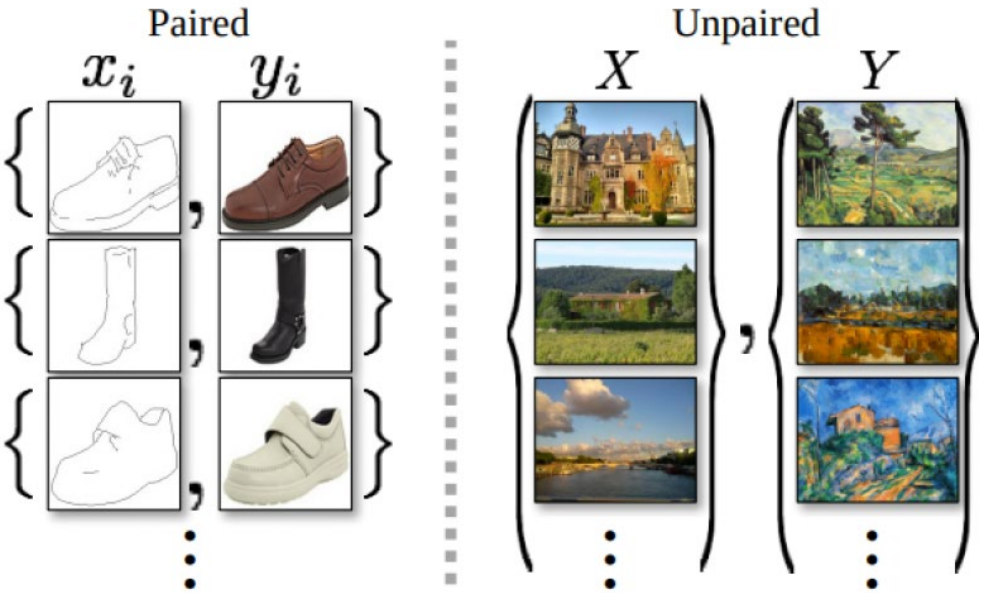
Domain A would just be algorithmically augmented images (turn color images to greyscale)

Domain B would be the unchanged images.

What is the difference between paired and unpaired image translation? List an example task for both kinds. What is cycle consistency and why is it useful?

In the paired case images from the two domains correspond directly to each other while in the unpaired case this connection cannot be made.

For example, greyscale – color is paired and portrait photos – cartoon characters are not.



To overcome this challenge two generators are used. One "F" to map A → B and another "G" to map B → A.

- Two generators $\mathcal{G} : A \mapsto B$ and $\mathcal{F} : B \mapsto A$

Core idea is that we demand cycle consistency

- $\mathcal{F}(\mathcal{G}(\mathbf{x})) \approx \mathbf{x}$ for $\mathbf{x} \in A$
- $\mathcal{G}(\mathcal{F}(\mathbf{y})) \approx \mathbf{y}$ for $\mathbf{y} \in B$

Additionally generators should leave images from their target domain unchanged!

MAE loss is used for CycleGANs in addition to the established adversarial losses (one for A→B and one for B→A),

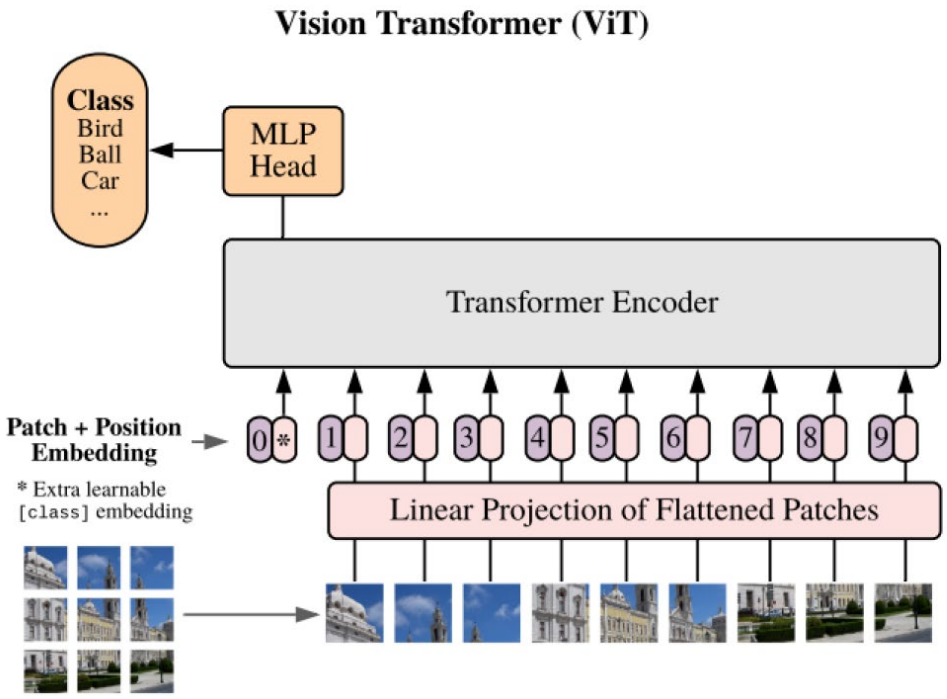
What is a transformer, and what are its inputs and outputs? How can images be made compatible with the input requirements? What is the theoretical advantage of transformers over convolutional neural nets?

Transformers are neural networks which have had a huge impact in deep learning in the last decade. Originally from the NLP domain, but now used in others as well.

The main feature of transformers is their attention mechanisms which enables comparing inputs to all previously seen tokens.

Transformers for images separate the image into tokens which are then used like the words in an NLP problem.

The encoding that the transformer produces are then used in a conventional MLP classifier.



The way CNNs work is heavily based on the fact that we are working with images which introduces **inductive bias** because of the assumptions made that inform certain design decisions.

When reducing this bias we might be able to get even better models!

Because Transformers are not designed for Images they have less of a specific bias. This means that they might learn decision boundaries not accessible to CNNS.

What is data bias? Name and explain three types. Explain the term protected attribute and list at least four of them.

Data bias is bias that is introduced directly from the data set we use. It can be due to the way something is measured or unseen confounding variables in the ground truth which we fail to observe.

Types of data bias:

- Historical bias (males as CEOs)
- Representation bias (only western countries included)
- Measurement bias (sensor that only works in certain conditions)
- Evaluation bias (using inappropriate evaluation metrics)
- Aggregation bias (false conclusions because of wrong population assumptions)
- Population bias (e.g. only CS students in data set)

Protected attributes:

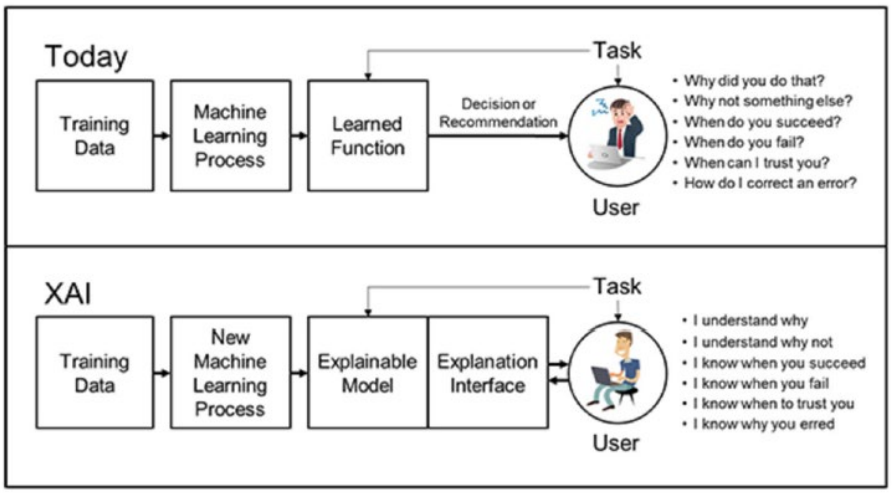
Protected attributes are properties of humans that cannot legally be discriminated by.

In the EU the se are:

- Sex
- Race
- Color
- Ethnic or social origin
- Genetic features
- Language
- Religion or belief
- Political or any other opinion
- Membership of a national minority
- Property
- Birth
- Disability
- Age
- Sexual orientation

Why should models be explainable? What does this mean? How can fairness be assessed? Why is in-processing for a running AI solution hard to implement?

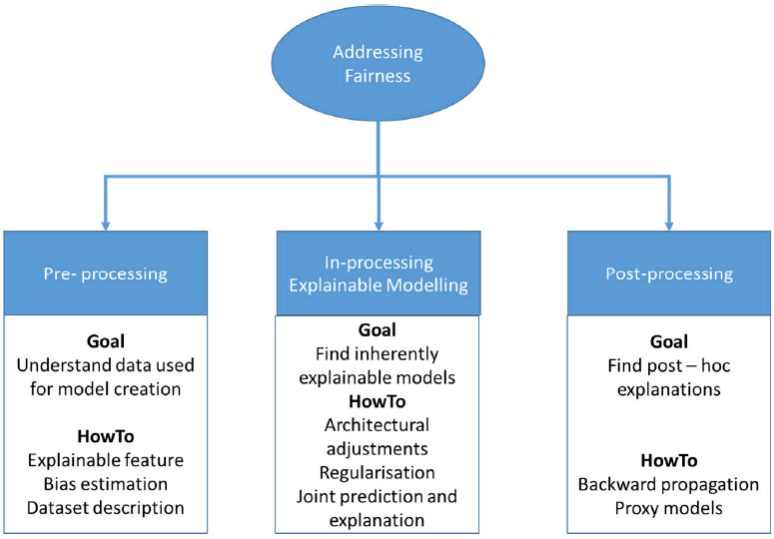
Model explainability is important to enable trust in AI models. Their adoption in critical areas of life can only be realized when explainability is established.



Key benefits are:

- User trust
- Bias identification
- Legal compliance
- Fairness

<u>Assessing Fairness:</u>



In-processing fairness is hard to implement because many models are extremely complex and black-boxes that no human can hope to understand currently. Model performance may suffer because of the changes made to achieve fairness.