Image processing for Earth Observation

4c

Tricks to train convnets

Devis TUIA

EPFL, fall semester 2025

École polytechnique fédérale de Lausanne

Content (6 weeks)

- W1 General concepts of image classification / segmentation
  Traditional supervised classification methods (RF)
- W2 Traditional supervised classification methods (SVM)
  Best practices

- W3 Elements of neural networks
- W4 Convolutional neural networks
- W5 Convolutional neural networks for semantic segmentation
- W6 Sequence modeling, change detection

Just a little extra

- Convnets overfit very quickly, they have millions (if not more) learnable parameters.
- With semantic segmentation (next week), this will become even worse (we will predict pixel responses)!
- Today we will briefly see 3 ways to prevent it:
  - Batch Normalization [Ioffe and Szegedy, 2015]
  - Dropout [Srivastava et al., 2014]
  - Data augmentation
- There are many more...

Tricks to avoid overfitting while training convnets

- Batch normalisation
- Dropout
- Data augmentation

A batch?

- A batch is simply a part of your training set
- CNNs are large models that need large numbers of observations to train
- The GPU cards we use are limited in memory
- At each batch, we pass one part of the training set and update the parameters (= stochastic gradient descent)
- How many samples in a batch depends on the size of your model (how many parameters to learn) and of your GPU.
- 1 epoch of the neural nework is over when the model has seen ALL the training data

Batch Normalisation what?

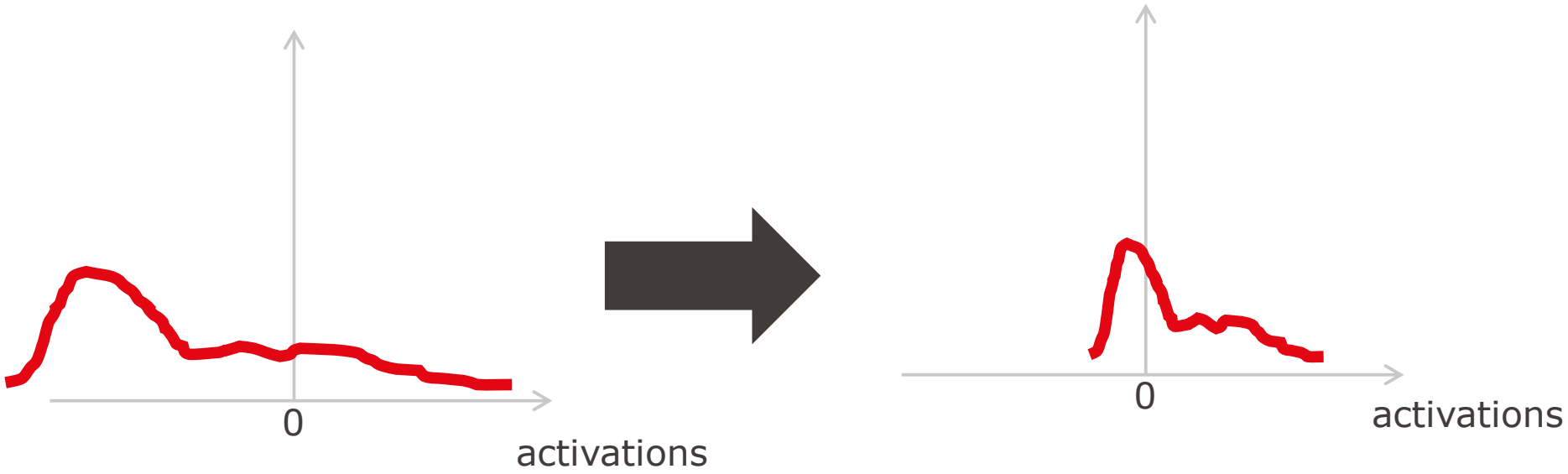
- Conceptually: *normalize all your feature maps with the statistics you observe within each batch*

$$BN(\mathbf{x}) = \gamma \circ \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

- γ and β are two learnable parameters (like those in the convolution filters) and are learned while training
- μ and σ are the mean and variance of the batch being considered

Batch Normalisation effect?

- As for data standardization in machine learning in general, BN modifies the values of the data.
- BN changes the activation maps so that they all have **zero mean and unit variance.**



Batch Normalisation
Why?

- **Normalizing is always good.** Processing inputs with similar scaling put all parameters on a similar level a-priori.
- Similar scaling also makes life simpler for the next layers, since you **combine inputs with similar values.** If one of your activation maps has values 100x bigger than the others, you must adjust for the differences in the learnable parameters. This slows down convergence and might need layer-specific learning rates (which we don't want).
- BN is a form or **regularization** (it avoids being too dependent on a single layer *"because it has high values"*). Regularization is always good in learning.

Batch Normalisation How?

- You can apply it everywhere (convolutional or fc layers)
- Usually it is applied before the nonlinearity.
- It is applied differently in training and testing
  - Training: as the formula (slide 5). We apply it at batch level and update the activations
  - Testing: we use the global mean and variances observed during training and update slightly*

*: For this, at each batch we keep updating a global average and variance with momentum, something like *global_mean = 0.9*current_global_mean + 0.1*batch_mean*

Tricks to avoid overfitting while training convnets

- Batch normalisation
- Dropout
- Data augmentation

Dropout what?

- The idea of Dropout is to “turn” off neurons at random during the forward pass.
- Given a dropout probability of *p:*

$$h' = \begin{cases} 0 & \text{with probability } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}$$

- *The normalization by (1-p)* is to keep the same expectation of the neuron with respect to the one with the neurons off: E[h’] = h

Dropout effect?

- The idea of Dropout is to “turn” off neurons at random during the forward pass.
- On a MLP it would look like this:

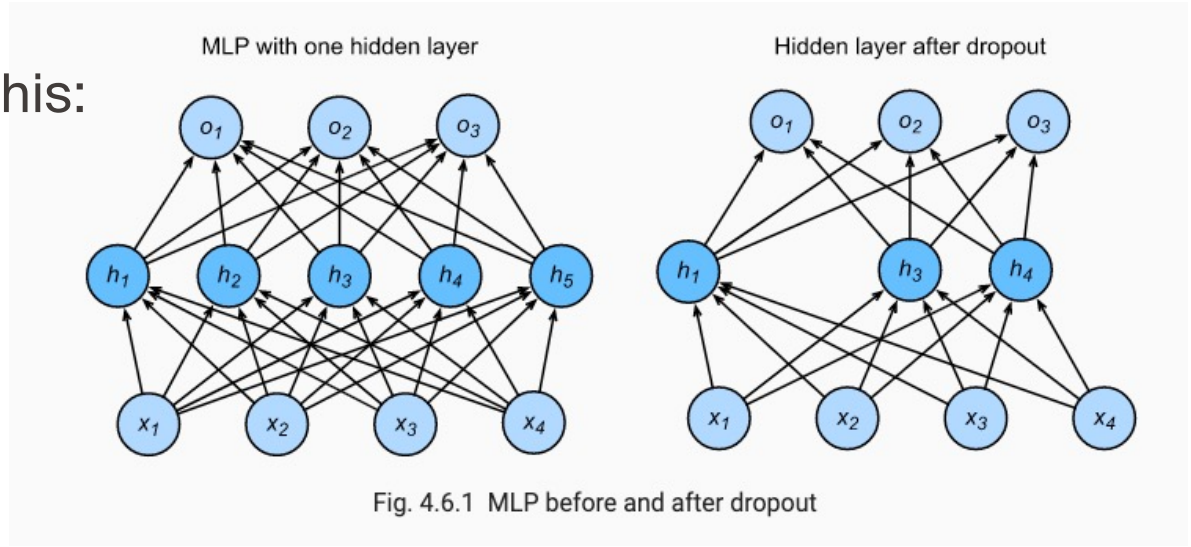


Fig. 4.6.1 MLP before and after dropout

- Which neurons are turned off changes every time the forward pass is used.

Dropout effect?

- This makes that neurons cannot co-adapt:
  - no preferential path can be learned (= no path is always taken)
  - slightly different models are learned at every path
  - the final model is a kind of average (p = 0 is used a test time)

- In the end it is not so different from ensemble models (remember Random Forests?)

Dropout
Why?

- As we said before, **regularization** is always good against overfitting
- Using **ensembles** regularizes, as the final outcome is the geometric mean of the single models, but
  - Training different models and then averaging them is computationally unfeasible
  - Given that CNNs have huge capacity, most of those models would not train well or require very extensive model fitting
- Dropout was the answer to that: to **approximately combine similar models during training** of a single one.

Dropout
How?

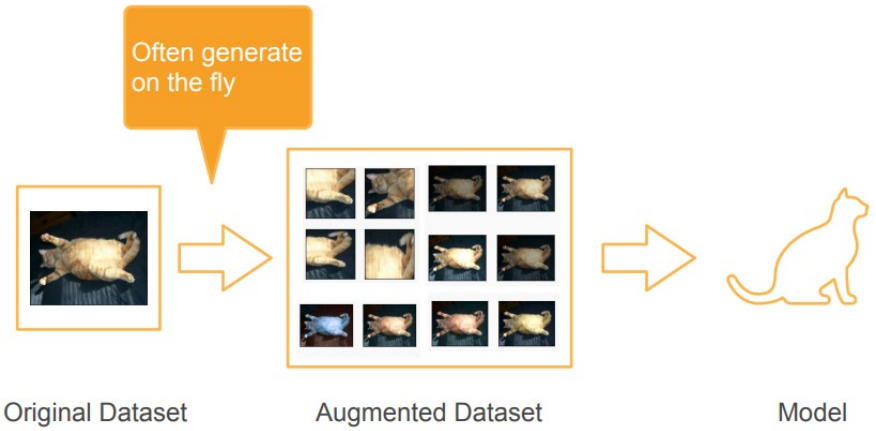
- It is generally only applied in fully connected layers. Why?
- The probability is a user-defined parameter
- It is applied differently in training and testing
  - Training: as the formula. We apply a different mask every time a forward pass is called
  - Testing: we don't use it.

Tricks to avoid overfitting while training convnets

- Batch normalisation
- Dropout
- Data augmentation

Data augmentation what?

- The idea of data augmentation is to create additional training examples by applying sensible modifications to the data you have
- In a nutshell: load training data, modify it slightly, use it for training
- When new data is loaded (at next epoch), apply a different modification



https://courses.d2l.ai/berkeley-stat-157/slides/3_14/15-Finetune.pdf

Effect

- Training with augmented data helps the model seeing more examples
- Seeing more variety helps generalisation
- If the augmentations make sense, the model can memorize more properties of the problem
  - Rotated images: invariance to rotations
  - Color changes: invariance to atmospheric effects
  - Jittering (noise injection): insensitivities to sensor noise
  - Random cropping: little invariance to image resolution
  - ...

Data augmentation why?

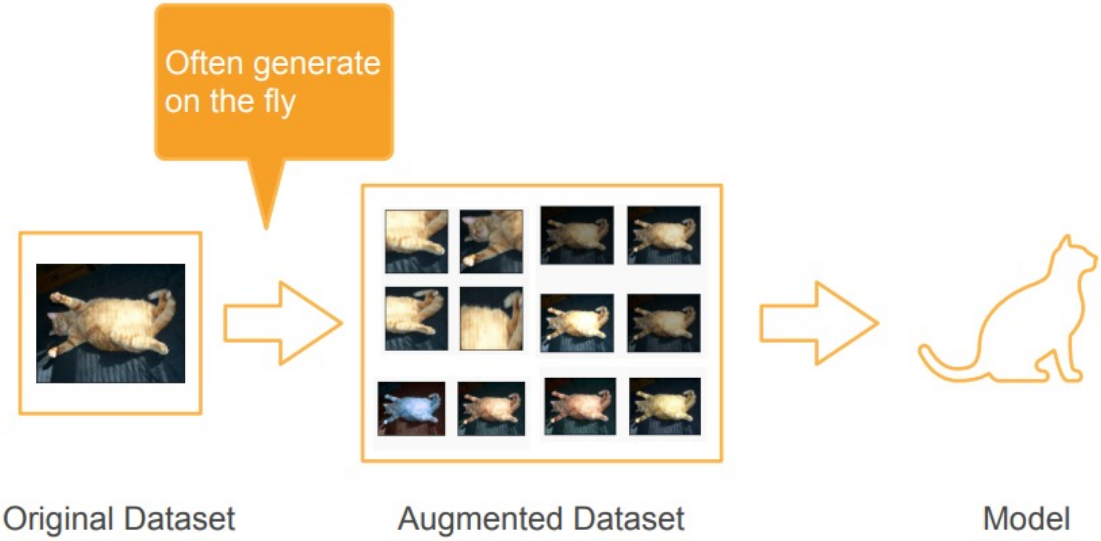
- State of the art neural networks consist of (b)millions of parameters.
- Training those parameters require proportional amount of data !!!
- More data = better training

- BUT

- Augmentation is ALWAYS to be applied with care (don’t exagerate, the artificial images need to remain “realistic”).

Data augmentation
How?

- Apply random modifications from a defined set while loading the training examples



https://courses.d2l.ai/berkeley-stat-157/slides/3_14/15-Finetune.pdf

Examples of augmentations

Cropping

Flipping

Examples of augmentations

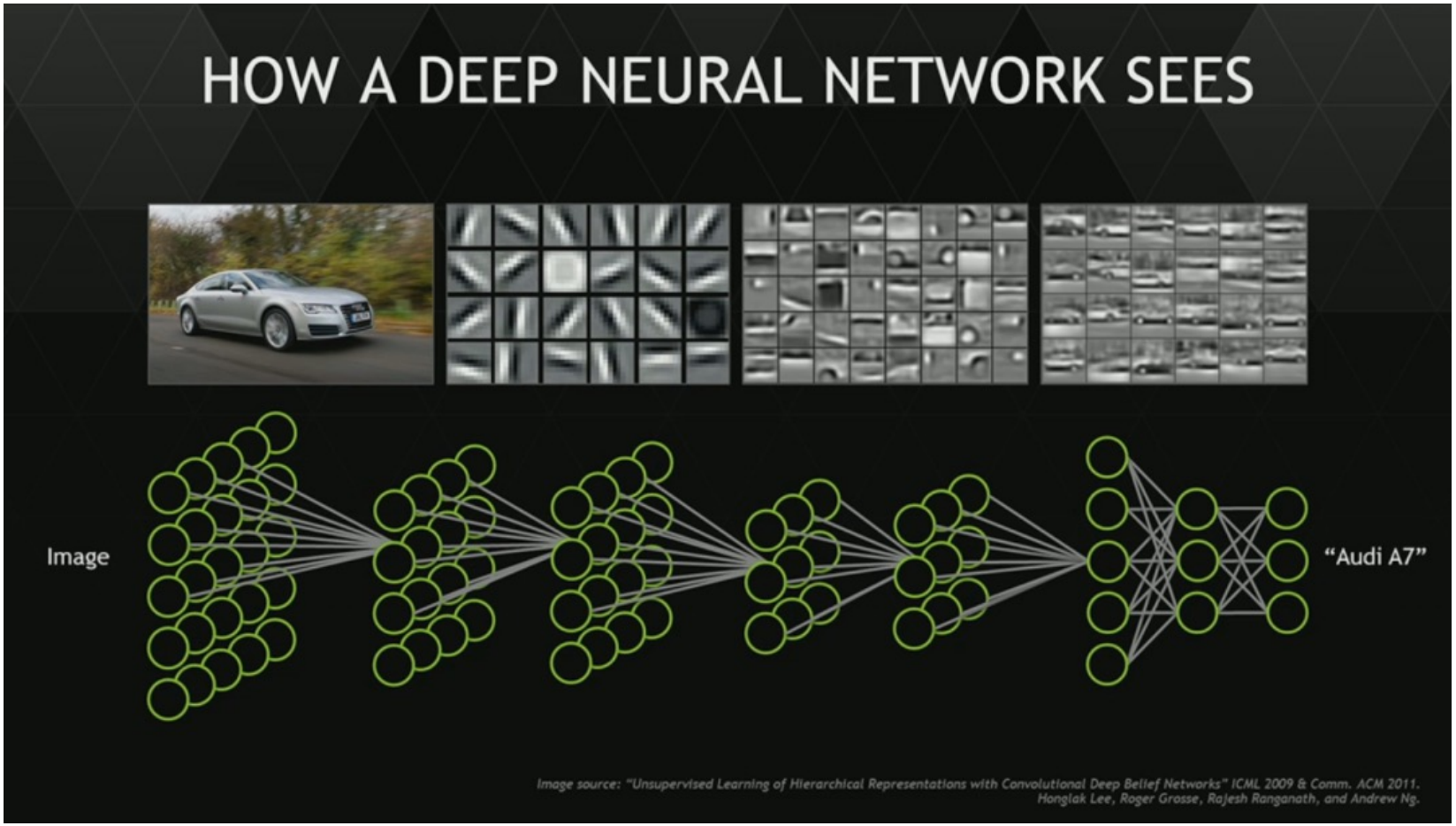
Color modifications

Combinations

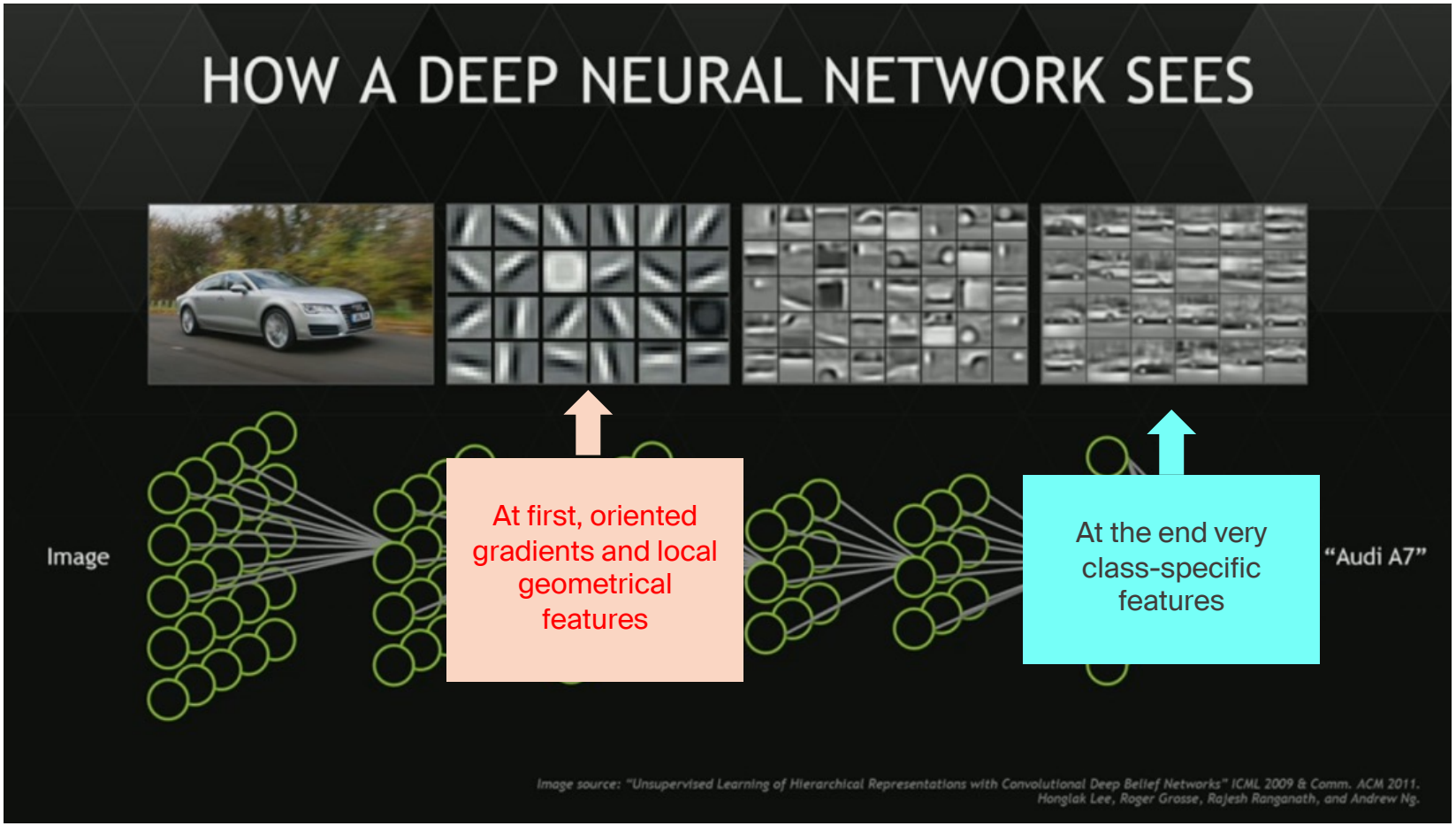
Ready to fine-tune!

- Now you are ready to tune your models!
- You can either learn them from 0 (from scratch) or start from an existing model learned from natural images. We call these models pre-trained
- We use the pre-trained model as a starting point and make it specific to our needs. We call this procedure **fine-tuning**
- In a nutshell:
  - you take a pre-trained model,
  - change the last layer (to predict the number of classes that you want)
  - Tune to your problem by training from there.

- Why does it work?



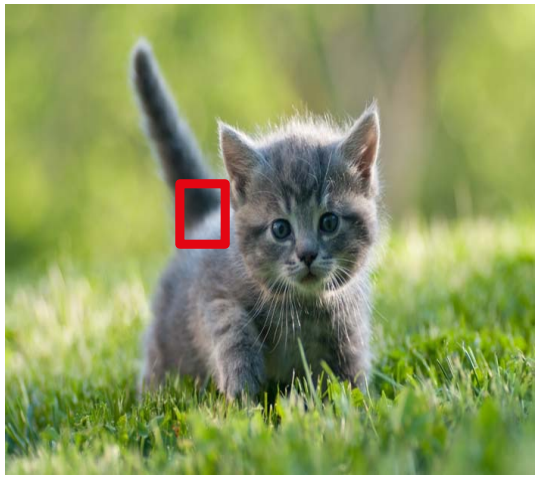
[Source: Nvidia (Lee), CES2015]



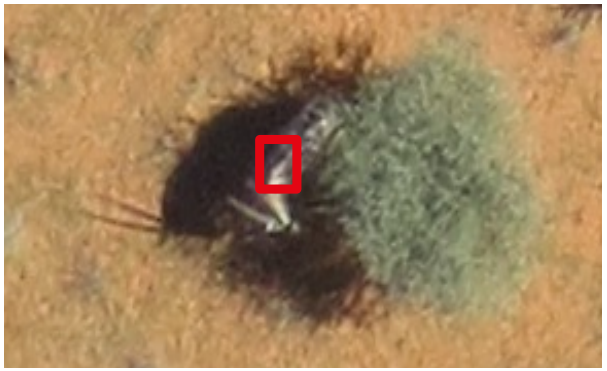
[Source: Nvidia (Lee), CES2015]

The early layer of convnets are quite generic!

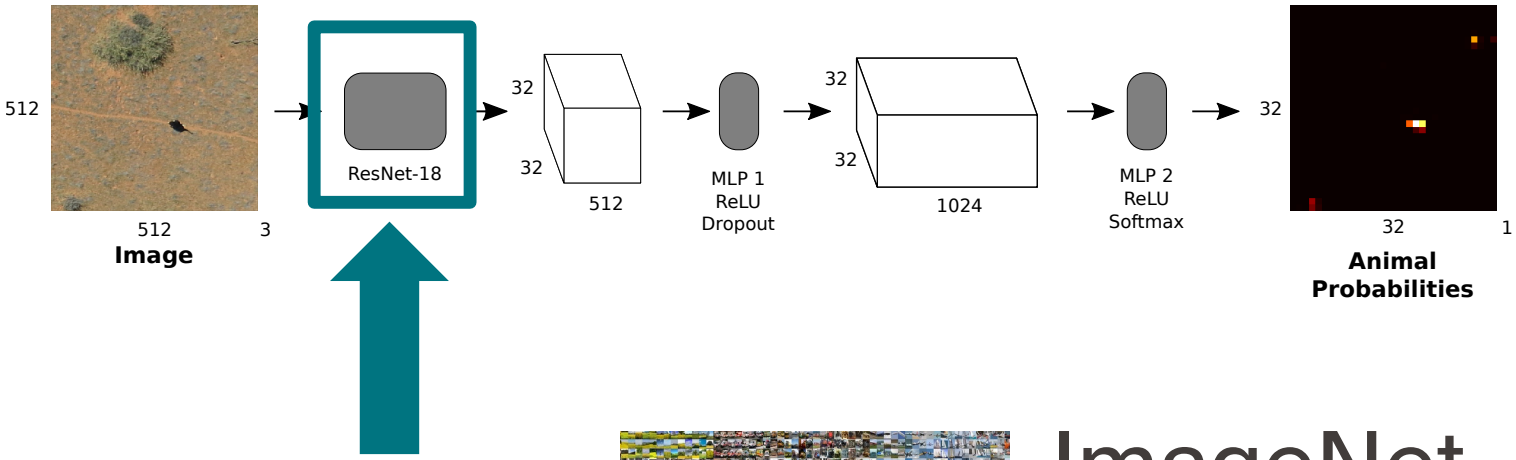
- Probably we don't need to learn the low level features all over again.
- So we can take the early features directly from ImageNet and only learn the specific ones.



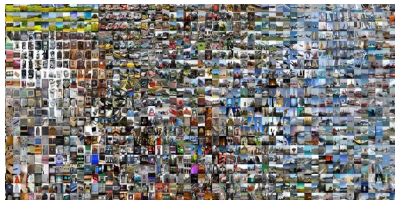
www.dierendokters.com



Using ResNet 18 trained on ImageNet



We use a pre model trained on ImageNet



ImageNet

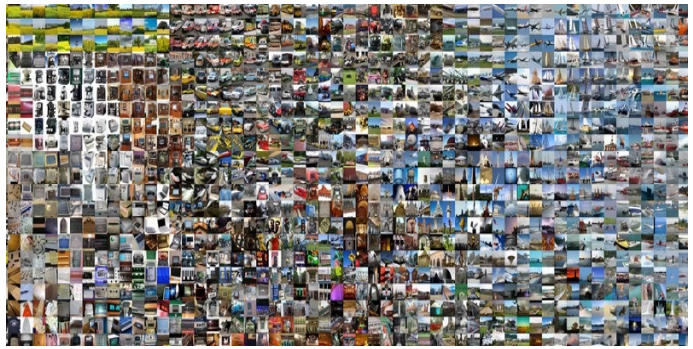
http://www.image-net.org

Natural images
1000 classes (person, car, cat, ...)
>14 Million images

B. Kellenberger, D. Marcos, and D. Tuia. Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning. *Remote Sens. Environ.*, 216:139–153, 2018.

Fine tuning a pre-trained model

- There are many such models around. Many of them are pre-trained on a large dataset called **ImageNet**
- They are called: LeNet, AlexNet, VGG, ResNet, EfficientNet, ...



ImageNet

http://www.image-net.org
Natural images
1000 classes (person, car, cat, ...)
>14 Million images

You are now a real convnet padawan

- Now overfitting fighting you can
- To train convnets ready you are

- Have fun with the exercise now you will