Deep Learning - AlexNet

Bernhard Kainz

AlexNet

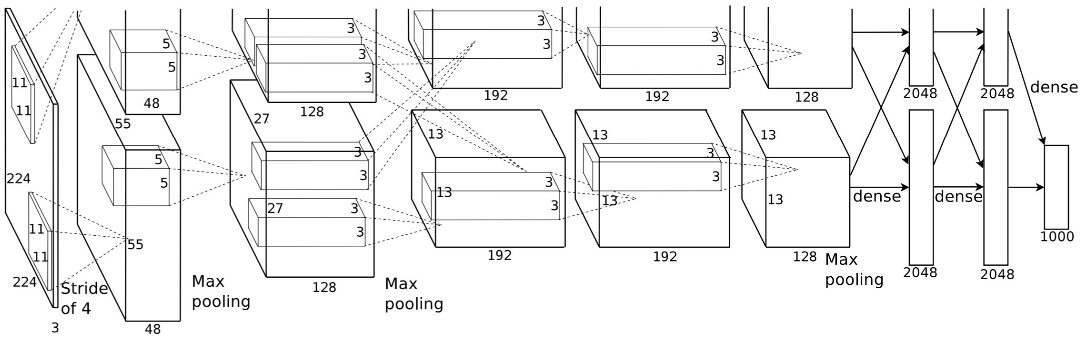


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

AlexNet from 2012 was the first large scale convolutional neural network that was able to do well on the ImageNet classification task
In 2012 AlexNet was entered in the competition, and was able to outperform all previous non deep learning based models by a significant margin, and so this was the convNet that started the spree of convNet research usage afterwards.

the basic convNet AlexNet architecture is a conv layer followed by pooling layer, normalization, covolution, pool norm, and then a few more conv layers, a pooling layer, and then several fully connected layers afterwards.
So this actually looks very similar to the LeNet network.
There are just more layers in total.
There are five of these conv layers, and two fully connected layers before the final fully connected layer going to the output classes.

Alexnet takes as input images of size 224 by 224 by 3. so these are 3 channel colour images
it uses 11 by 11 filters at stride 4. other people showed in the meantime that it isn't necessary to have such large filter kernels.
we have a couple of fully connected layers of size 4096 and finally the last layer is FC8 going to the soft max, which is going to the 1000 ImageNet classes.

if you look at this AlexNet diagram, it looks kind of like the normal convNet diagrams that we've been seeing, except for one difference, which is that it's, you can see it's kind of split in these two different rows
or columns going across. The reason for this is mostly historical.

AlexNet was trained on old GTX580 GPUs that only had 3 gigs of memory.

So it couldn't actually fit this entire network on these cards, and what they ended up doing, was they spread the network across two GPUs.

So on each GPU you would have half of the neurons, or half of the feature maps. And so for example if you look at this first conv layer, we have 55 by 55 by 96 output, but if you look at this diagram carefully,
you can see that, it's actually only 48 depth-wise, on each GPU, and so they just split the feature maps, directly in half.

What happens is that for most of these layers, for example conv one, two, four and five, the connections are only with feature maps on the same GPU, so you would take as input, half of the feature maps that were on the the same GPU as before and you don't look at the full 96 feature maps for example.

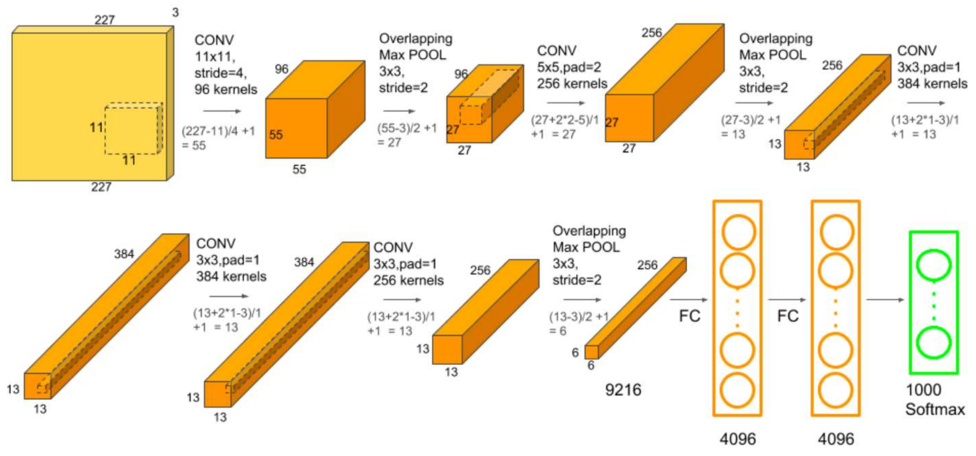
You just take as input the 48 in that first layer.

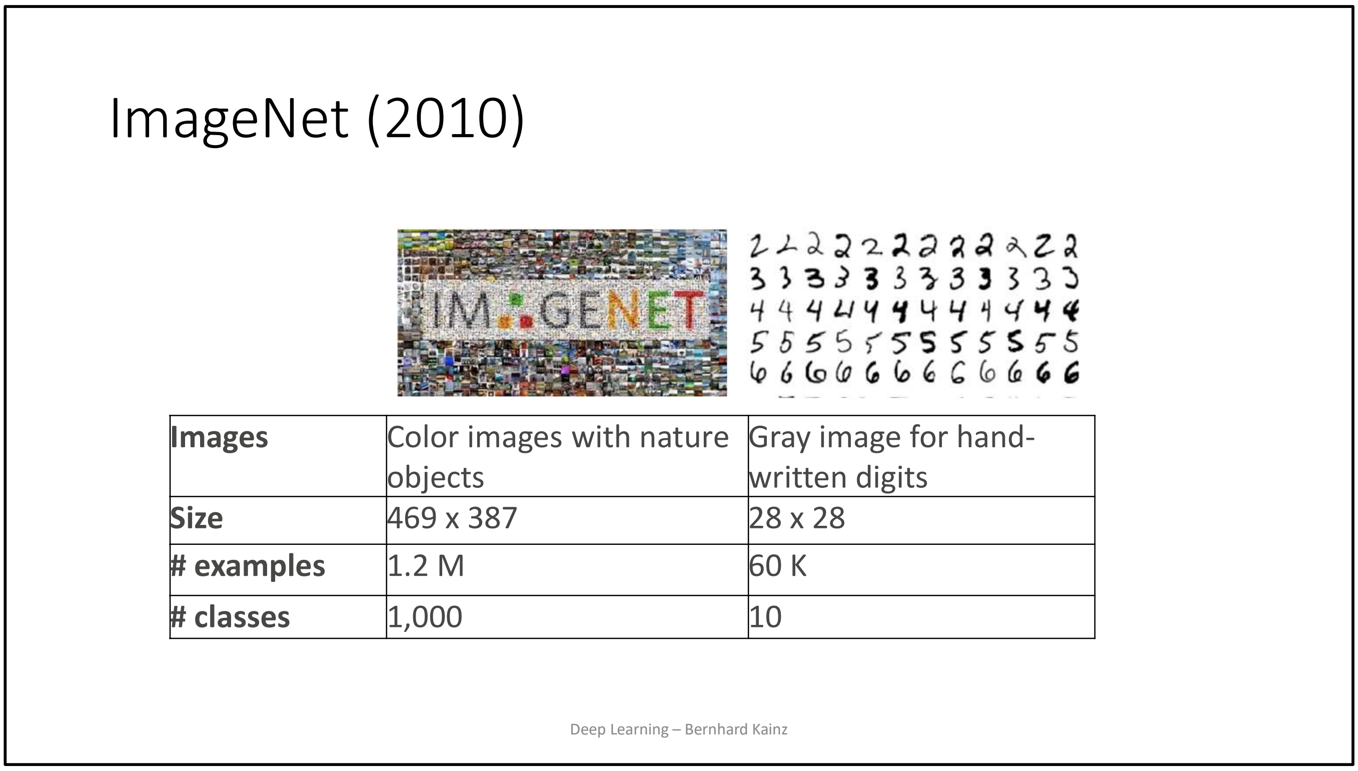
And then there's a few layers so conv three, as well as FC six, seven and eight, where here are the GPUs do talk to each other and so there's connections with all feature maps in the preceding layer.

so there's communication across the GPUs, and each of these neurons are then connected to the full depth of the previous input layer.

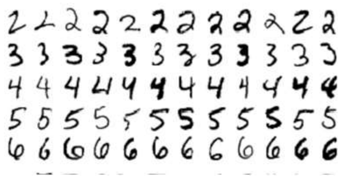
Nowadays you get this overhead for free, for example with pyhtorch's DataParallel features.

AlexNet


227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax



ImageNet (2010)



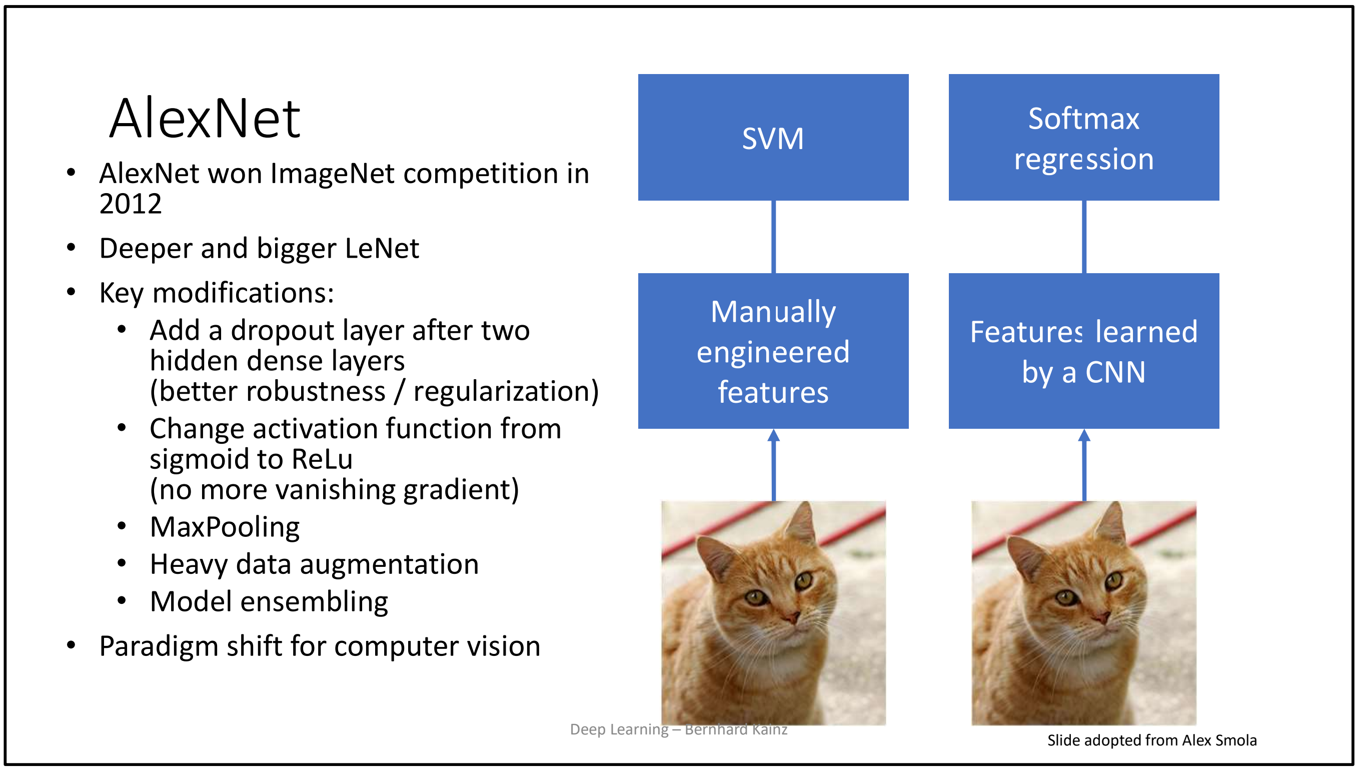
Images	Color images with nature objects	Gray image for hand-written digits
Size	469 x 387	28 x 28
# examples	1.2 M	60 K
# classes	1,000	10

Deep Learning – Bernhard Kainz

let's look at the data.
imagenet came out in 2010 and it was a big data set at the time with 1.2 million examples, thousand classes

compare that to 60 thousand samples, ten classes for MNIST

also the resolution was considerably bigger so it went from 28 by 28 to 469 by 384 dimensions and the images were in three channels namely red green and blue, whereas before that we had great scale so that changed things a lot.



AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Key modifications:
  - Add a dropout layer after two hidden dense layers (better robustness / regularization)
  - Change activation function from sigmoid to ReLu (no more vanishing gradient)
  - MaxPooling
  - Heavy data augmentation
  - Model ensembling
- Paradigm shift for computer vision

SVM
Softmax regression
Manually engineered features
Features learned by a CNN

Slide adopted from Alex Smola

until 2012 around that time when Alex net1 won the imagenet competition the default strategy for solving computer vision problems was to go and pick manually engineered features and apply an SVM in the end.
This was replaced by features that were learned automatically followed by a softmax function.

but AlexNet wasn't just a bigger and better LeNet there were a number of other key changes

one was drop out regularization which allowed people to design much deeper networks as you move to deeper networks

of course just regularizing with regard to the input doesn't help so much, you need to also regularize the inner structure of the network so this is this introduction of regularization applied to all the layers
of the network or at least in multiple places whenever you use dropout

whereas otherwise you would just smooth things with regard to the input

the second thing was really rectified linear units replacing the otherwise common sigmoid non-linearity by just the max between X and 0 which had as a consequence that the gradient would no longer vanish because you had at least 1/2 of the space where the function was the identity

Another thing was max pooling which replaced average pooling and then the result of that was that now features were rather a bit more shift invariant because you could now move your attributes a little bit and max pooling would still pull the relevant attributes through

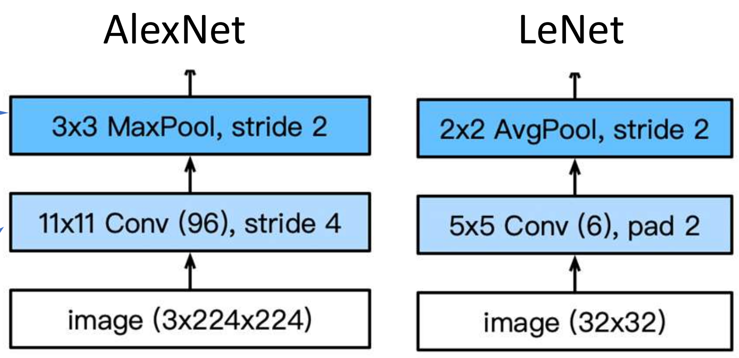
To eventually win this challenge, heavy data augmentation was used, so cropping, shifting rotation of the inputs together with model ensembling, where you train multiple versions of the same model and average their results.

so this led to a paradigm shift in computer vision and after computer vision well that's then when people went to speech recognition, natural language processing, text generation and a lot of other things.

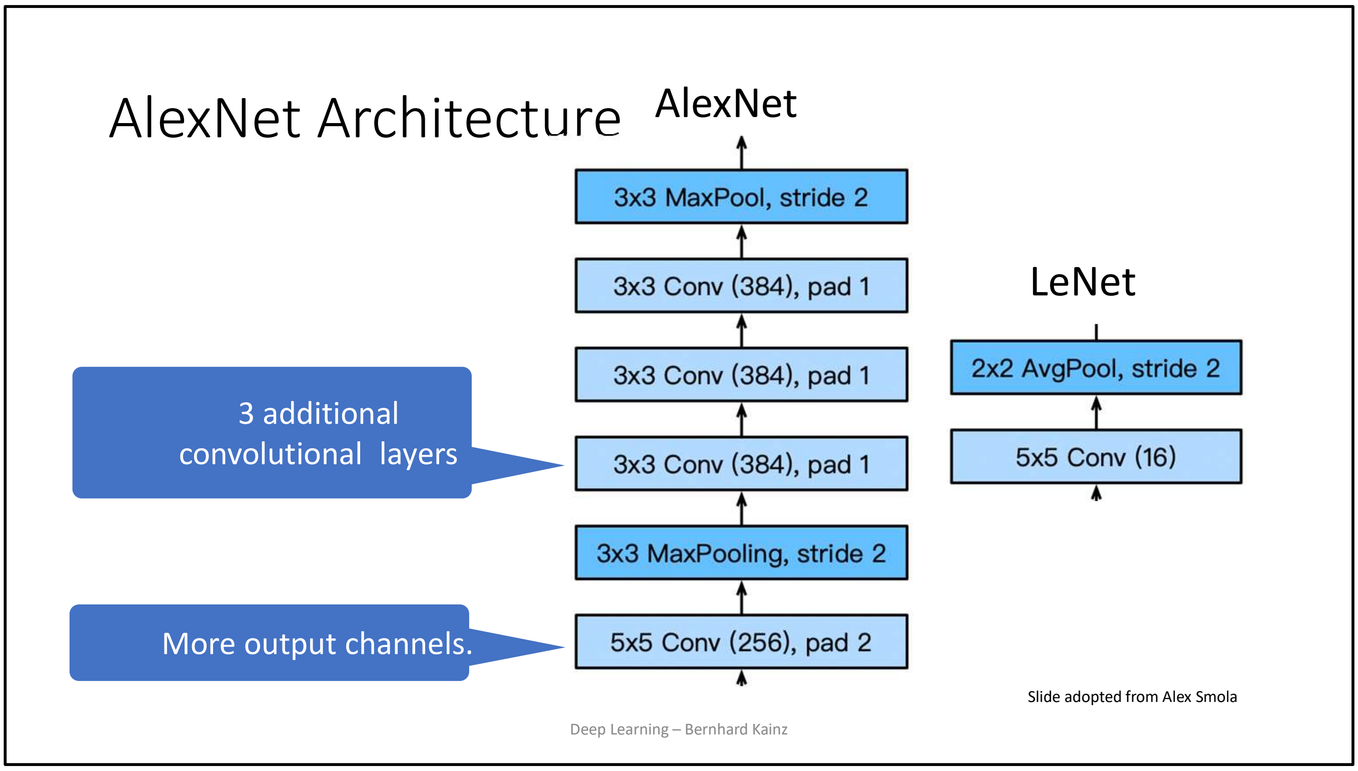
AlexNet Architecture

Larger pool size, change to m
pooling

Larger kernel size, stride
because of the increased
image size, and more outpu
channels.



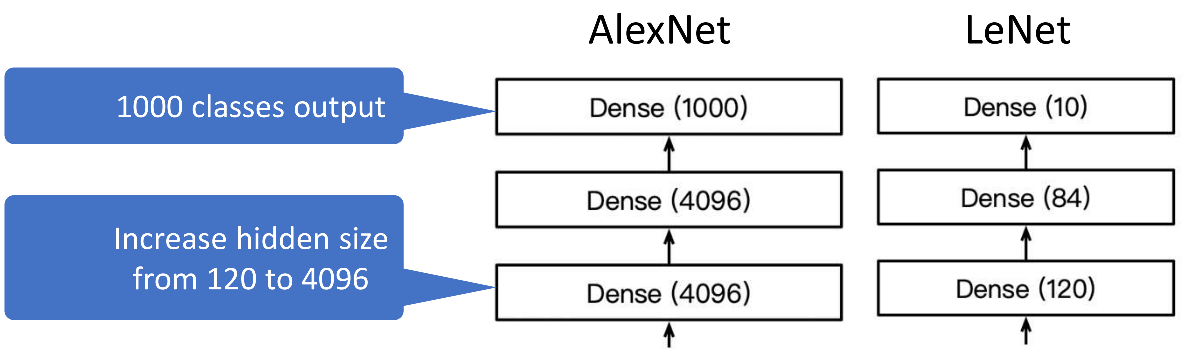
Slide adopted from Alex Smola



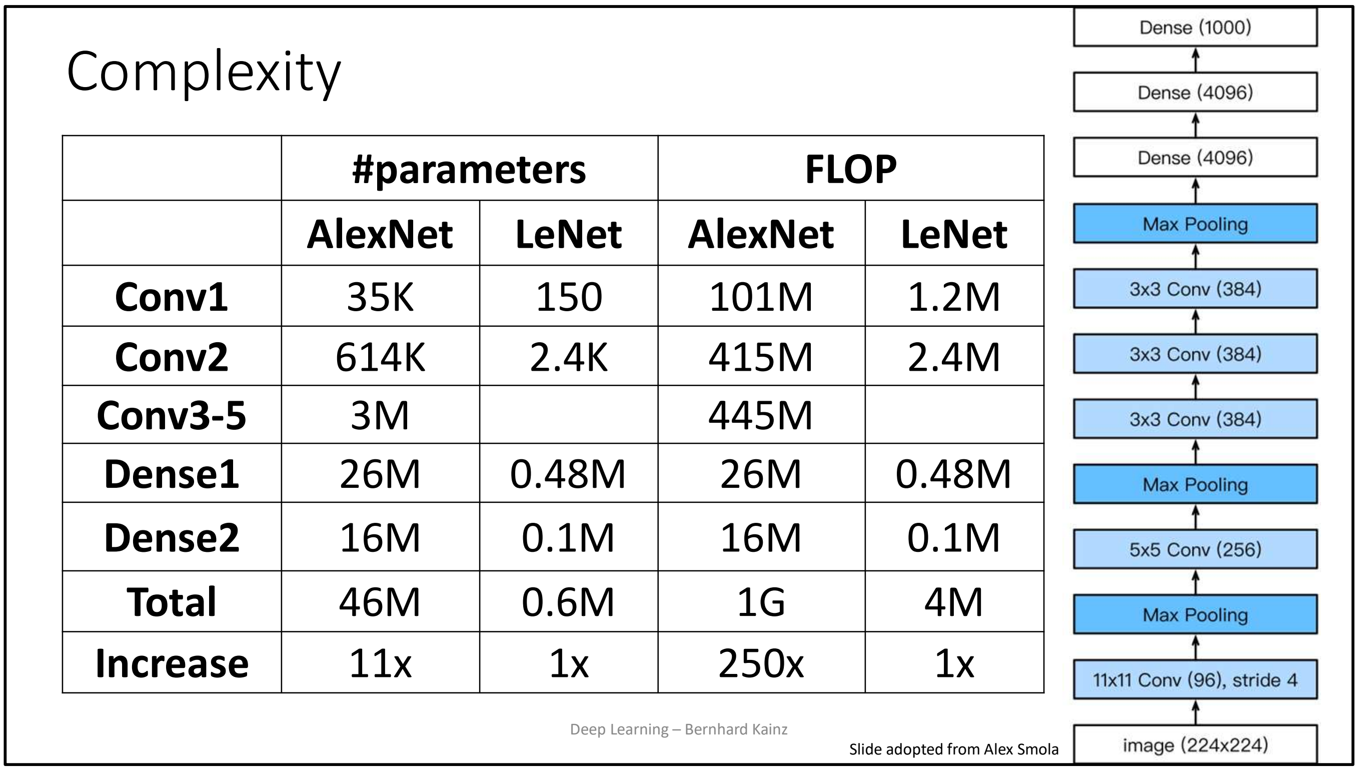
AlexNet Architecture

Slide adopted from Alex Smola

AlexNet Architecture

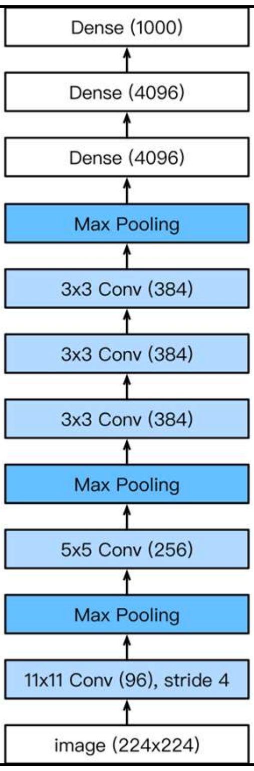


Slide adopted from Alex Smola



Complexity

	#parameters		FLOP	
	AlexNet	**LeNet**	**AlexNet**	**LeNet**
Conv1	35K	150	101M	1.2M
Conv2	614K	2.4K	415M	2.4M
Conv3-5	3M		445M	
Dense1	26M	0.48M	26M	0.48M
Dense2	16M	0.1M	16M	0.1M
Total	46M	0.6M	1G	4M
Increase	11x	1x	250x	1x



Slide adopted from Alex Smola

if you look at the complexity of such networks, well, AlexNet is a lot more complex than LeNet-5

In terms of computation it's 250 times more expensive; in terms of parameters only ten times more

and this was another key change that the trade-off between computation and memory changed quite a bit and alexnet is actually known for being rather extreme in terms of its memory usage

so nowadays that ratio would have been probably even much more skewed towards compute because compute devices have become a lot faster and therefore people like to exploit that.

BK2

demo

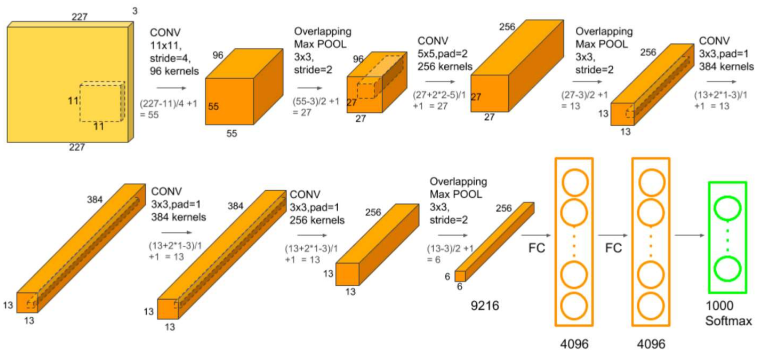
Silicon Valley: Season 4 Episode 4: Not Hotdog (HBO)
https://www.youtube.com/watch?v=pqTntG1RXSY

I have been looking hard for a demo of AlexNet but really all you would see is a slightly worse labelling of the data than provided in the test set as ground truth. Instead I show you what people thought at the time what's suddenly possible.

Slide 145

BK2 Bernhard Kainz, 22/09/2020

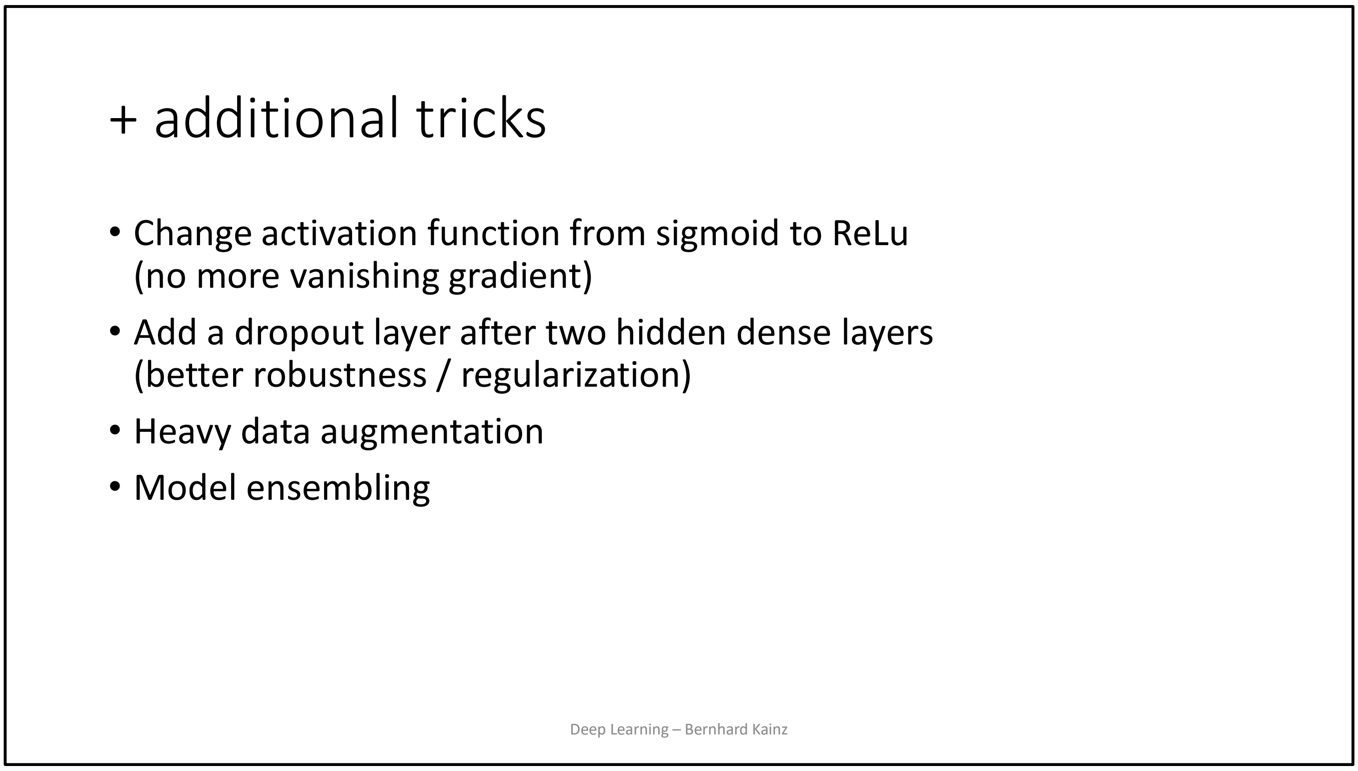
```python
class AlexNet(nn.Module):

    def __init__(self, num_classes=1000):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x
```



227
3
11
11
227
CONV 11x11, stride=4, 96 kernels
(227-11)/4 +1 = 55
96
55
55
Overlapping Max POOL 3x3, stride=2
(55-3)/2 +1 = 27
96
27
27
CONV 5x5,pad=2 256 kernels
(27+2*2-5)/1 +1 = 27
256
27
27
Overlapping Max POOL 3x3, stride=2
(27-3)/2 +1 = 13
256
13
13
CONV 3x3,pad=1 384 kernels
(13+2*1-3)/1 +1 = 13
384
13
13
CONV 3x3,pad=1 384 kernels
(13+2*1-3)/1 +1 = 13
384
13
13
CONV 3x3,pad=1 256 kernels
(13+2*1-3)/1 +1 = 13
256
13
13
Overlapping Max POOL 3x3, stride=2
(13-3)/2 +1 = 6
256
6
6
9216
FC
4096
FC
4096
1000 Softmax

https://github.com/pytorch/vision/blob/master/torchvision/models/alexnet.py

This is how you can implement Alexnet in pytorch. It also shows a consolidated network diagram since nowadays there is no need to split it over two GPUs anymore.
Talk through code



+ additional tricks

- Change activation function from sigmoid to ReLu (no more vanishing gradient)
- Add a dropout layer after two hidden dense layers (better robustness / regularization)
- Heavy data augmentation
- Model ensembling

Deep Learning – Bernhard Kainz

in order to get the best numbers they also did an ensembling of models and so training multiple of these, averaging them together and this also gives an improvement in performance.