Machine Learning for Human Vision and Language

Lecture 1:
Principles of deep learning in artificial networks

Ben Harvey

Welcome to Machine Learning for Human Vision & Language
This is a course from the faculty of social sciences and humanities, where we will focus on sensory and cognitive processing.
In the first half of the course we will look at deep convolutional neural networks, a type of machine learning network whose design is inspired by the function of neurons in the brain, and the structure of networks among these neurons. We will look at how these artificial and biological networks are related, with the specific example of networks for object recognition and how this works in the human visual system.

The second half of the course will look at networks for language processing. These parts have some similarities and some differences, but it is a lot less clear how language processing works in the biological systems so modern computational linguistics relies heavily on artificial neural network models. To understand what information these use to model language, you will look more at the mathematical and logical structure of language.

First, let's go through an outline of the course so you know what to expect and what we expect from you.

Course goals

- Explore the relationship between cognitive sciences and AI
- Focus on deep learning in artificial machine learning networks and comparison to biological systems
  - Which biological processes do deep networks imitate?
  - What is missing in artificial networks?
  - What might make AI/machine learning more like biological intelligence/learning
- Become familiar with the use of AI in cognitive science research
- Build some deep learning networks to do human-like tasks

Some of this lecture content is complex, and you have very different backgrounds.
But it will be examined, and you must pass this exam to pass the course.
So we will go slowly, start from first principles and avoid assuming any knowledge.
Please ask questions when you don't understand: the explanations and discussion that result are very valuable to the class format.

Assessment

- In groups of 2, complete two ‘learning based’ lab assignments to build simple deep learning systems to solve computer vision and language analysis tasks.
  - You are welcome to work on these exercises with one other group.
  - Develop written reports answering questions about your work.
  - Grades depend on depth and completion of these reports (10% each).
  - You will get considerable feedback as you progress, and your teachers will grade you as you go. You can use this feedback to improve your work and grading.
- In groups of 2, complete two ‘challenge based’ lab assignments, extending and assessing what you have learned.
  - Work on these without input from your teacher or other groups.
  - There is no need to work on these during class time.
  - Grades depend on depth and completion of these reports (10% each).
- Your understanding of lectures and reading assignments will be assessed in two exams (30% each).
- You are required to average a passing grade (5.5) **averaged across the two exams** to pass the course. Students scoring above 4.0 on the **whole course** qualify for a repair exam.

3

In these assignments you will learn about deep learning by coding to build networks, and those of you with some programming experience often find this relatively straightforward.
But that is not what we are grading. You will see that a lot of the assignment’s questions ask you how you understand and interpret things. This is because we are also teaching from the background of cognitive science.

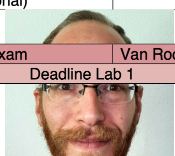
So, you are all good students. We expect almost everyone to get high grades on the learning based assignments, mostly 8s and 9s.
The challenge-based assignments help us distinguish between good and great.
In the final grades for these assignments, most groups will get grades in the 6s (acceptable) 7s (good) and 8s (great), a few will get 9s (excellent).

Similarly in the exams, we will ask you to explain things and we will grade you based on how clear and complete your explanations are. You are taking a Masters degree, and we expect very good answers for the highest grades.

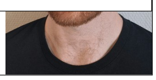
You need to show us you know and understand the answer. To show us you understand, you need to be able to explain it.

For the lab assignments, there is no second attempt to improve your grade, so work hard on them and submit really great answers.

Date	Time	Format	Teacher	Room
03/09	15:15-17:00	Lecture 1	Harvey	KBG-ATLAS
05/09	13:15-17:00	Lab 1	Van Rooij	BBG 201, 209, 214
10/09	15:15-17:00	Lecture 2	Harvey	KBG-ATLAS
12/09	13:15-17:00	Lab 1	Van Rooij	BBG 201, 209, 214
17/09	15:15-17:00	Lecture 3	Harvey	KBG-ATLAS
19/09	13:15-17:00	Lab 1	Van Rooij	BBG 201, 209, 214
24/09	15:15-17:00	Lecture 4	Harvey	KBG-ATLAS
26/09	13:15-17:00	Lab 1	Van Rooij	BBG 201, 209, 214
01/10	15:15-17:00	Exam study question time (optional)	Harvey	KBG-ATLAS
03/10	11:00-13:00	Midterm exam	Van Rooij	Educatorium ALFA
07/10	23:59	Deadline Lab 1		Brightspace
03/12	13:15-15:00	Exam viewing (BOTH EXAMS)	Harvey/ Abzianidze	Educatorium Theatron
14/01/ 2026	17:00-20:00	Repair exam	Both	Educatorium ALFA







Daan van Rooij
Lab 1 coordinator

4

Our lectures will normally be on Wednesdays. We do this one on a Friday to prepare you for the lab assignments.

On Friday afternoons you will have lab classes coordinated by Daan van Rooij, who you can contact with administrative questions about Lab Assignment 1.
In these Friday lab classes classes, we expect you to work together and get help with your ‘learning based’ lab assignments.
Outside class time, you won’t be able to get help with your work from your lab teachers.

16 hours of class time should be enough to finish these and begin the challenge-based assignments.

So if you are falling behind, make sure your work is almost complete with the learning-based assignments by the last lab class so you can get help in that class.

On the Wednesday after the our last lab class, we have an optional class where you can come to ask questions about your exam study. This is in the time when we would usually have a lecture.

After this, we have a midterm exam focussing on the content of the first 4 lectures and part 1 of the reading list.
The deadline for submitting your assignments comes a few days after that, so you can use that time to finish your challenge-based assignment. You should submit both of your assignments for the same deadline, though there will be different

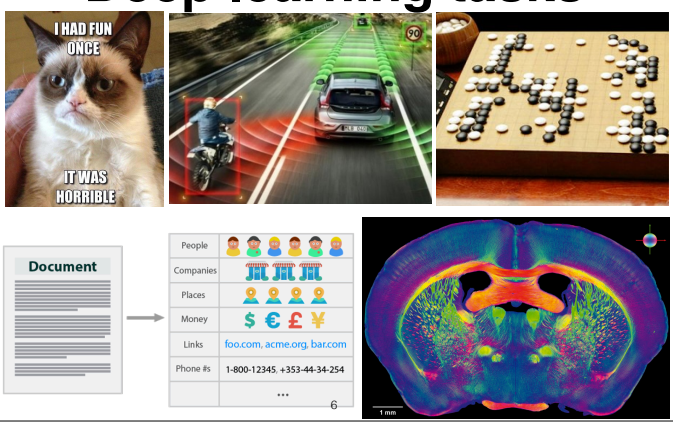
---

Why deep learning?

- Deep learning is particularly useful in tasks that are:
  - Hard/impossible to describe using formal mathematical rules
  - BUT easy for humans to perform
    - Intuitive or automatic
- Simulation of neural computation

5

Deep learning tasks



6

Image processing
Game opponents in complex games
Natural language processing
Simulation of biological neural systems

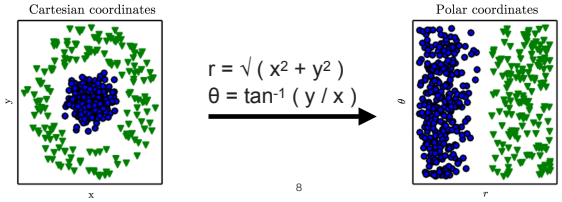
Deep learning approach

- Learn from experience (machine learning)
- Process inputs through a hierarchy of concepts
  - Each concept defined by its relationship to simpler concepts
  - So, build complicated concepts out of simpler concepts

This is what a human is doing when learning about the world
SO the main conceptual inspiration for deep learning is the brain. As we will see, the design of deep learning systems has also followed the function of the brain increasingly closely.
However, artificial neural networks simplify the processes involved considerably for computational efficiency, and many of the engineering choices made are chosen to allow the neural network models to run on computer systems like graphic processing units.

Representations & features

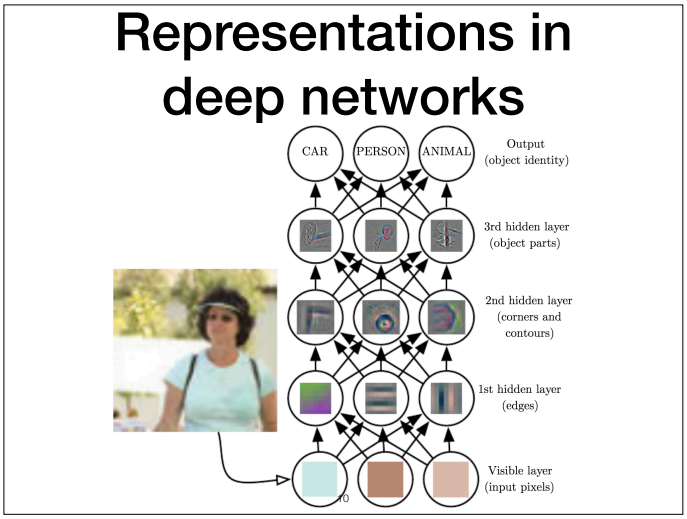
- Machine learning performance depends on the **representation** of the case to be classified
  - What information the computer is given about the situation
- Each piece of input information is known as a **feature**
  - The same feature can be represented in different formats
    - Often easy to convert between formats
  - The chosen format strongly affects the difficulty of the task



A simple task like this can be solved by choosing the right set of features, with minimal learning necessary.
However, for many tasks, it is hard to know which features or formats of the input are important in determining the output.
And these may be high-level features that need to be extracted first.
Deep learning aims to determine which formats of their representations are optimal for solving their task, through experience

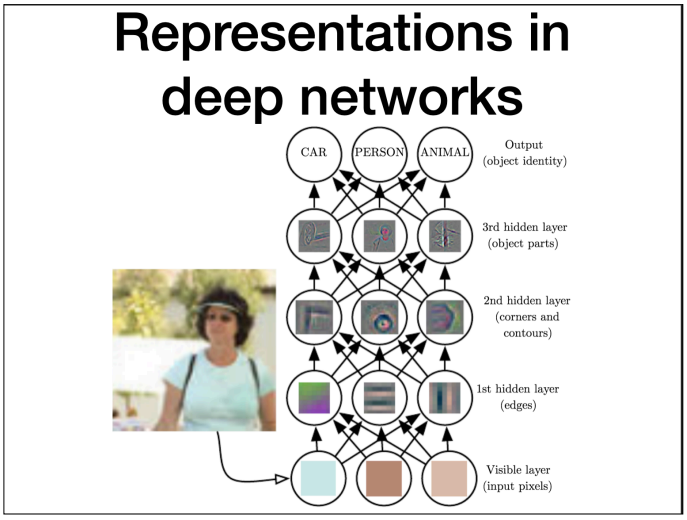
Representations in deep networks

- Useful features may need to be transformed or extracted first
- So deep networks have multiple representations
  - Each is built from an earlier representation
- This can:
  - Transform information to a different format before learning its links to the output
  - Extract complex patterns of information through integration of simpler information
- Essentially multiple steps in a program
  - Each layer can be seen as the computer's memory state after executing a set of instructions
  - Deeper networks execute more instructions in sequence
- Just like a computer program, the individual steps are generally very simple
  - Complex outcomes emerge from interactions between many simple steps



Here we can see how this abstract description might work in an oversimplified example of object recognition.
The first layer just takes the colour of each pixel.
This is transformed to the edge representation in the next layer by learning common relationships between these pixels.
The edges are then transformed to corners and contours by learning relationships between the edges.
The next layer finds object parts by learning common patterns of corners and contours.
These object parts are then transformed into whole object representations by learning which patterns of object parts correspond to which object type.

We will return to the example of object recognition many times
It's an excellent example of a process that is intuitive and automatic, but hard to formalise or program.
It is also very useful for computers to do, so we can find images on the internet without a human labelling their content.
Finally, object recognition is a problem that has now been solved, so we can really see how the result works.



A quick note on notes.
All of these slides will be available online, but you will find I use little text on my slides. This works better in class, but is hard to study from.
I deal with this in two ways. First, my online slides also contain notes with a fairly complete description of what I say. This is very easy to make notes on and study from. Please note that your other lecturers won't give you extensive notes.

Second, I record these lectures and also put these movies on Blackboard. But I will say here that watching videos online is a terrible substitute for attending classes with your peers, as remote learning has shown us over the last few years. Students who only watch our recorded lectures generally learn far less than students who come

to class, and get lower grades. Computers are very distracting.
Also, watching a video is a terrible way to study the content of the lectures. It's very useful if you need to miss a class. It's a fine starting point for studying. But to remember things and be able to explain them back to us in exams, active processes like reading and writing your own notes are far more effective.

What is a deep network?

- A learning network that **transforms** or **extracts** features using:
  - Multiple **nonlinear** processing units
  - Arranged in multiple **layers** with
    - **Hierarchical organisation**
    - Different levels of **representation** and abstraction

12

Note that this definition does not specify 'machine' learning.
In this course, we will also look at biological neural networks like the brain, which are also deep networks

Lectures 1-4

- Lecture 1: Principles of deep learning in artificial networks
- Lecture 2: Deep learning in biological neurons and networks
- Lecture 3: Feedforward visual processing
- Lecture 4: Recurrent visual processing

13

So, we're going to start by looking at artificial networks, a machine learning system.
We will then compare these to the human brain's biological neural networks that these artificial networks aim to imitate.
In my last two lectures, we will look at the stages of visual processing involved in object recognition and other visual tasks in both systems.
This will show what is missing in current artificial networks and how researchers are increasingly building deep networks that fill these gaps.
To begin, let's look at what machine learning is, starting artificial networks from the start.

What is machine learning?

- The field of science that 'gives computers the ability to learn without being explicitly programmed' (Arthur Samuel, 1959)
- 'A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.' (Tom Mitchell, 1998)
  - Expressed in organisational terms, not cognitive terms

So this field has been around for a long time
But Samuel's definition of machine learning uses the word 'learn' without any definitions
It also uses 'computer', which means the same as 'machine' here.
Finally, the computers HAVE been explicitly programmed: to 'learn'.
So this just says machine learning is whatever allows machines to 'learn'.

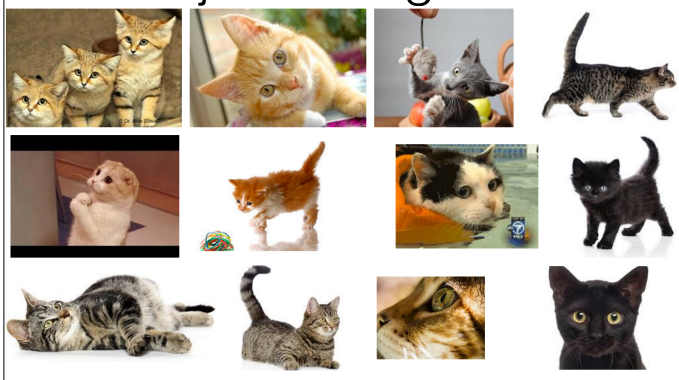
Mitchell provided a more formal definition of this learning: performance improves with experience.
By describing the fundamental operation in terms of inputs (experience), outputs (performance) and goals (tasks), this definition avoids implying the computer should do the 'learning' like a human would.
When humans learn, we can see by observing humans that their performance improves with experience, so that's what we look for in

What is a deep network?

- A learning network that **transforms** or **extracts** features using:
  - Multiple **nonlinear** processing units
  - Arranged in multiple **layers** with
    - **Hierarchical organisation**
    - Different levels of **representation** and abstraction

This is a very broad definition, so we will use an example to see what it looks like
The example we will use in the first half of the course is object recognition.
This has been a major goal for deep learning in recent years, and is now largely solved, so we can investigate in depth how this works.
Object recognition may sound like an easy problem for computer vision and our own brain because it is effortless and automatic for humans.
But…

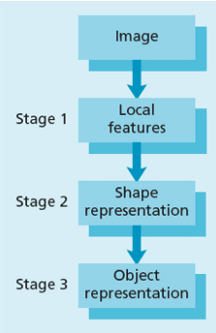
Object recognition



Why is it so difficult?

The identity of any object has little relationship to its impression on a camera's sensor or the eye's retina.
Here we see the result of a google image search for pictures of cats.
This used Google's artificial deep network trained for object recognition.
For this network and also for human vision, objects can be recognised from different viewpoint and sizes, in different positions, and with different lighting conditions.
Also, examples of the same class of object often look very different.
So we can't recognise an object directly from its impression on the eye or camera sensor

The 20th century view of object recognition

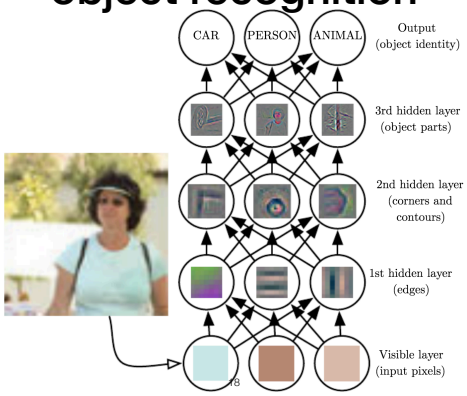


- Stage 1 builds a representation of local image features.
- Stage 2 builds a representation of larger-scale shapes and surfaces.
- Stage 3 matches shapes and surfaces with stored object representations-recognition.

17

Even in the 1980s it was already clear that some multi-level, hierarchical approach was needed to achieve object recognition.
After local features like edges were detected, a later stage would need to integrate and compare this information to detect larger-scale features like shapes. And these larger scale features would need to be integrated and compared to detect objects that were consistent with those shapes.

The 20th century view of object recognition



18

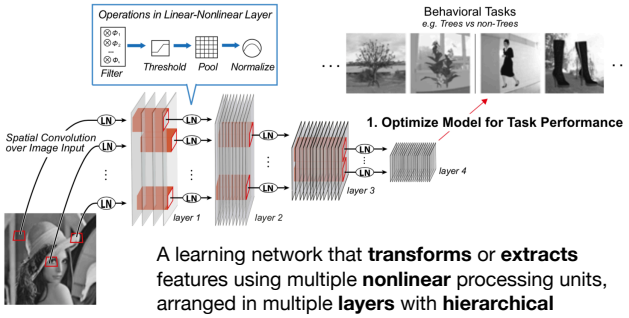
This was essentially the example we looked at earlier, here with the input at the bottom.

This has many similarities with a deep convolutional neural network, and is a good way to start thinking about how they work.
However, there are important differences between deep convolutional networks and what we see here.
Most importantly, the 2nd hidden layer is shown here to straightforwardly respond to corners and contours: straightforward combinations of edges.
Likewise, the 3rd hidden layer is shown to respond to object parts that could in turn be combinations of corners and contours.

Indeed, many objects are built from parts, so simplified parts that we recognise from all angles might let us build an object viewpoint-independent object representation.
However, no one has ever made a program that can do this for a large set of different objects.

It seems that the features considered in a model like this are too human.
When is a feature a corner and when is it a curve? Is there something in between? Can we define a corner, edge or surface so rigidly?
Essentially, all of these pre-defined steps tend to limit the network to recognise specific examples, rather than generalise to all possible objects in a class, which is the goal here.

A deep network for object recognition



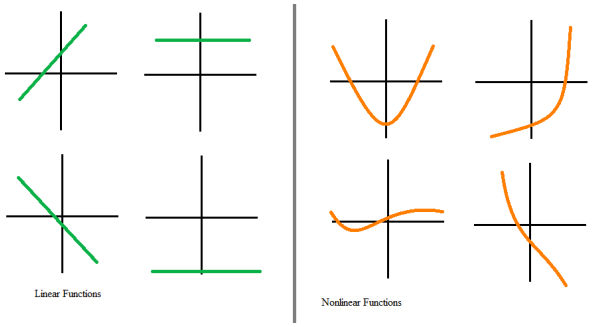
A learning network that **transforms** or **extracts** features using multiple **nonlinear** processing units, arranged in multiple **layers** with **hierarchical organisation** and different levels of **representation** and abstraction. 19

Here is a deep network which fulfils all the criteria in our definition.
It takes an input image and transforms its features to extract the class of object the image contains.
It is arranged in multiple layers, with one feeding into the next, forming a hierarchy. This is all much like the 20th-century idea.
The first layer represents the image pixels, with minimal abstraction, while the last layer captures object identity, which is highly abstract for a computer system.

But what happens to get from one to the other is very different from the 20th-century view.
The middle network layers do not respond to concrete concepts that are easy for us to think about, like corners and object parts. Instead they respond to whatever transformation of features is most beneficial for subsequently determining object identity.
This transformation of features is not easy for a human to conceptualise, as we will see.
So, let's take a short break, and then look at how this is done.

TAKE A BREAK HERE:
The last part of this definition is 'nonlinear processing units'.
But what does nonlinear mean, why is that necessary, and how is it achieved here?

Nonlinear functions



Y=A*X+B **Y≠A*X+B**

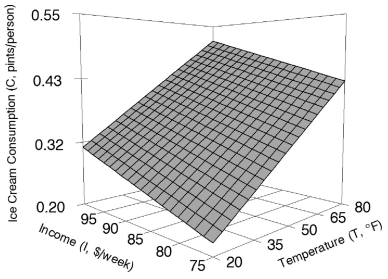
(Y is any other function of X)

20

In a linear function, the output (Y) of the function is simply the input (X) multiplied by a constant (A) and then added to another constant (B). The multiplier can be positive, negative or zero.
In a nonlinear function, there can be any other relationship between X and Y.
There must still be a relationship, Y is still a function of X, i.e. Y changes with X in some predictable way.
So we can see that non-linear functions can do a lot of things that linear functions can't.

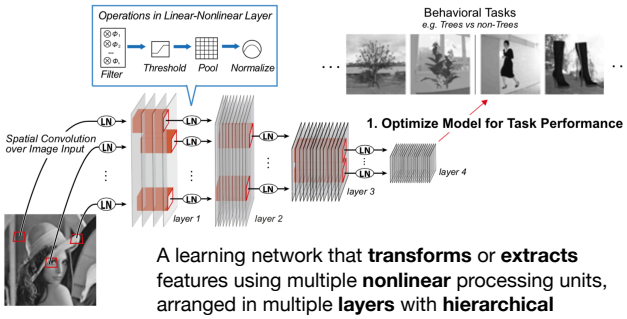
Why nonlinear functions?

$$Y = B + A_1*X_1 + A_2*X_2 + \ldots + A_p*X_p$$



But in the context of deep networks, there is a more important problem with linear functions.
The many layers of a deep network repeatedly perform functions on the output of previous stages. By doing so, more and more features of the input affect the output.
This example, going together only two inputs, gives an idea of the problem linear functions will face.
Joining multiple linear functions (by addition or multiplication) always results in a linear function of those multiple inputs.

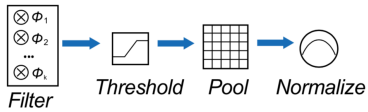
A deep network for object recognition



A learning network that **transforms** or **extracts** features using multiple **nonlinear** processing units, arranged in multiple **layers** with **hierarchical organisation** and different levels of **representation** and abstraction.

In this network, the inputs are the brightnesses of each image pixel.
There is no way these can be multiplied and summed together to give the likelihood this is an image of a tree.
Because, as we have seen, there is remarkably little relationship between an object's identity and the image it produces on the camera sensor.
Indeed, any operation that can be done with only linear functions of the input can be straightforwardly described by formal mathematical rules, so is not a good use for deep networks.

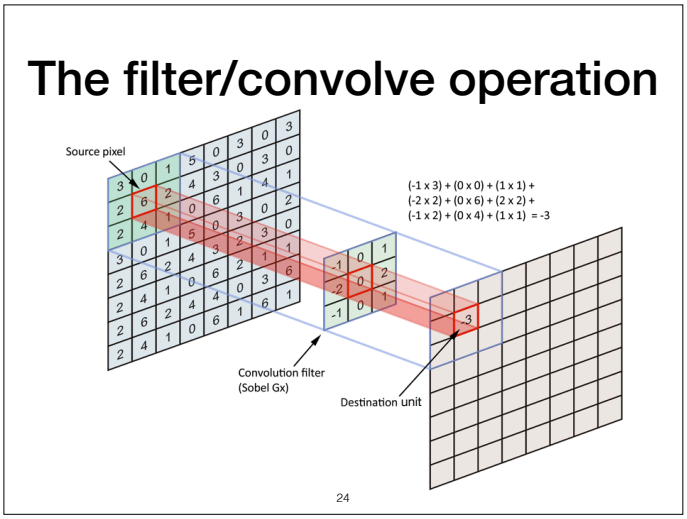
Operations in Linear-Nonlinear Layer



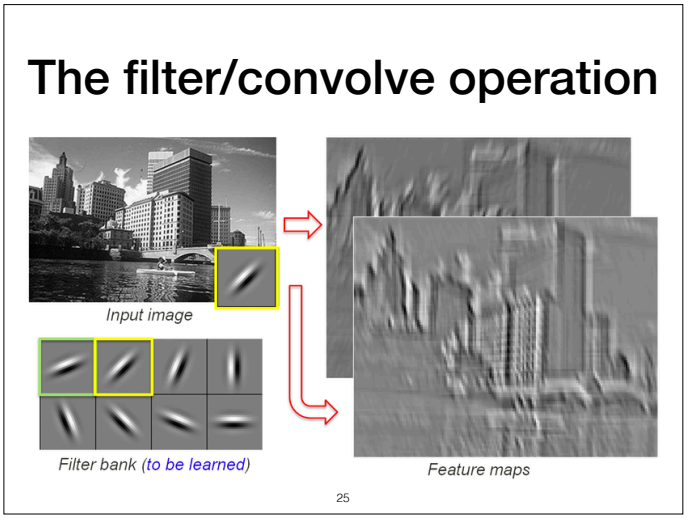
A learning network that **transforms** or **extracts** features using multiple **nonlinear** processing units, arranged in multiple **layers** with **hierarchical organisation** and different levels of **representation** and abstraction.

In our example, we have a complex nonlinear function with four operations or processing steps. These are filter, threshold, pool and normalise. These are very important to understand, so let's look at these steps in turn.

The output of one operation feeds into the next. The repeating sequence of these four operations effectively forms the layer.
The output of ALL these operations together forms the input to the next repetition of these operations, the next layer.
Note that this is described as a linear-nonlinear layer, which may be a little confusing.
The nonlinear function threshold is very important, but filter and normalise functions are linear. Pooling is also nonlinear, but optional.

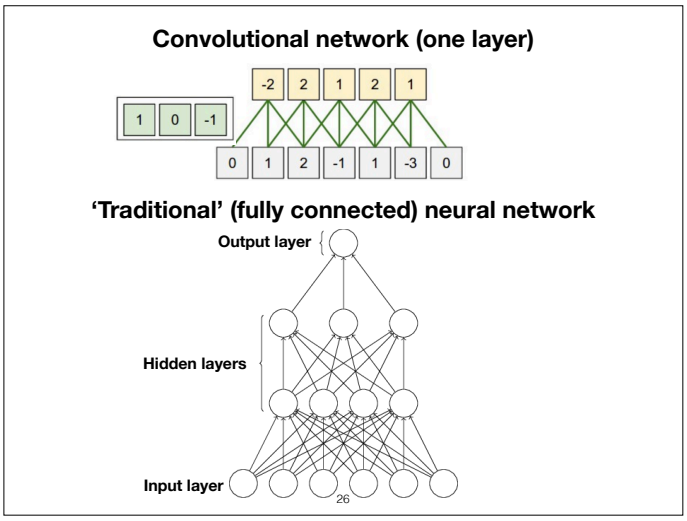


The filter or convolve operation is perhaps the most computationally important.
At the first processing layer, the input is simply the brightness of each pixel.
The convolution step looks for a pattern in a group of neighbouring pixels that corresponds to the convolution filter.
For this filter, this would be dark on the left (low numbers) and light on the right (high numbers).
In convolution, the weights in this filter are multiplied by a group of input pixels with a particular position and the products of these values are summed.
The result gives the match between the filter and a small part of the input image.
If the source pixels follow this filter pattern (dark on the left, light on the right), a high value will result. If the input area is all the same brightness, the result will be zero. If the source pixels are opposite to the filter (lighter on the left) the result will be negative.
Here, the match is poor: the pixel lightness on the left of the input source area is higher than on the right, so the output destination unit gets assigned a low value.
Then, the filter is moved by one pixel in the input image to give a value for the next output destination unit.
All source positions are multiplied by the same filter to fill in all the destination units. The convolution function is usually implemented using a single matrix operation over the whole input image, which processors can compute very efficiently.



The filter we just used is only one example to illustrate the principle.
In fact, a large set of filters are used in parallel to produce multiple maps of where their filter patterns are seen, often called feature maps.
Each ‘pixel’ shown in each of these feature maps is no longer a pixel in the image: it quantifies the match between the filter and that position in the input image.
This is then the activity of a neural network unit or artificial ‘neuron’
Here we see a set of eight larger filters with a range of different orientations.
These have been set manually in this example, because we know edge detectors with different orientations are an important early stage of computer and human vision systems.
But in real deep learning systems, the pattern of weights within these filters are actually determined by machine learning, though in the first layer the

result is an edge detector much like this.
The size of these filters will also determine the spatial scale of the feature we can detect. This size is set manually as a parameter of the network layer.



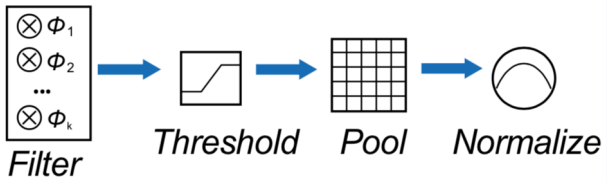
So, how is a convolutional network different from other types of neural network?
Here we see a 1-dimensional convolutional network.
Activation of each upper layer node depends on the product of the same filter (on the left) convolved with a spatially limited set of lower layer nodes.
In an earlier design, a fully-connected neural net, each node in a higher layer is connected to all nodes in the previous layer, with no constraint on the spatial spread of links between them. Again, the pattern of weights within these links is learned.
In a convolutional network, results of convolution with different filters are represented in different feature maps. But in a fully-connected neural net the input to a higher layer node comes from all low nodes in the previous layer. There is then no need to test a filter at all positions. So instead of mapping an input onto a feature map of nodes in the higher layer, each single node in the higher layer is a complete analysis of activity in the WHOLE lower layer, with no spatial representation at all.

A side effect of this is that the number of dimension in the input is irrelevant in a fully connected network, so we can simply view each network layer as 1-dimensional.

The constraint on the spatial spread of links between layers is particularly useful where the input has meaningful spatial relationship between neighbouring nodes or input pixels.
The most obvious examples of meaningful spatial relationships are in image processing, where the input layer units are image pixels and the feature maps each give the spatial distribution of the features described by the filters.
In the image example, the layers and filters are 2-dimensional because the input image is 2-dimensional. Here we are looking at 1-dimensional examples. 1-dimensional networks are more useful in linguistics, where the relationships between neighbouring letter or sound in determine which word they make and relationships between neighbouring words in a sentence determine what the sentence means.

The threshold/rectification operation

Operations in Linear-Nonlinear Layer



Y = f(X) = max(0, X)

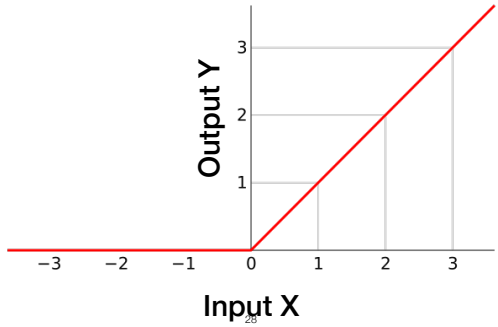
27

The nonlinearity is introduced by the threshold or rectification operation.
The goal of this operation is to only activate the output feature map if its value reaches a certain level, or threshold.
If we use filters that have a mean of zero, the threshold is typically zero.

The threshold/rectification operation

an activation function using a rectified linear unit (ReLU)

Y = f(X) = max(0, X)

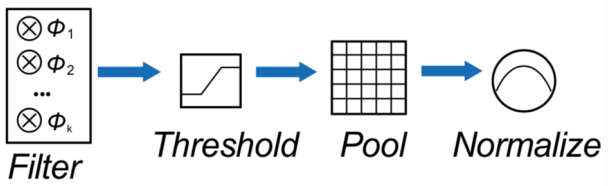


28

So, for values below zero in the feature map, after convolution, the output of the operation is zero.
For input values above zero, the output equals the input.
So this introduces a simple nonlinearity around zero.
Other 'activation functions' can be used to map input to output, but this is the most common in deep networks.
It is so common that the operation implementing the activation function in deep networks is typically just called ReLU, or a rectified linear unit.

The pooling operation

Operations in Linear-Nonlinear Layer



29

As a result of the filter operation, the response of each unit depends on several neighbouring inputs. So the units after filtering respond to a certain area of the input image, and the activation of neighbouring units will often be similar.
After several filter steps, each integrating inputs over an area, each unit will respond very similarly to an extensive area of the input.
So neighbouring units are representing very similar information.
The pooling operation therefore downsamples the units to improve computational efficiency.

The pooling operation

Feature Map

6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

Max-Pooling

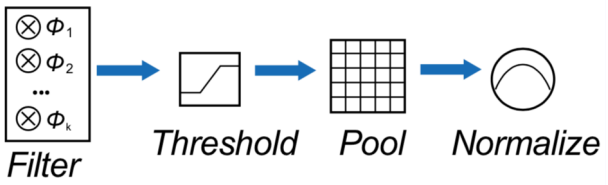
6	8
9	

30

This pooling operation is typically a simple ‘max’ operation, taking the maximum of a square of 2x2 neighbouring units of the feature map in this example.
So the pooling operation discards some data in favour of computational efficiency.
As computers and deep network implementations become faster and more efficient, it is becoming be less necessary to have pooling steps. This generally improves network performance but reduces speed.

The normalisation operation

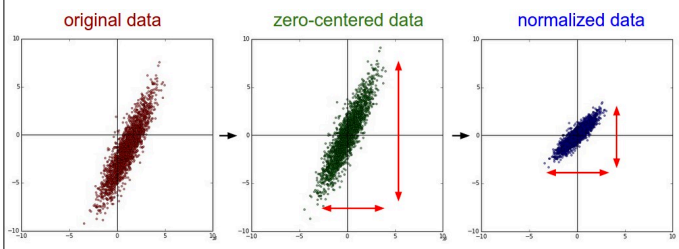
Operations in Linear-Nonlinear Layer



31

The threshold and pool operations use max functions.
As a result, even if the convolution filter has a mean of zero, by the pool stage we have a mean activation above zero.
Furthermore, the range of activations levels can be very different between feature maps, effectively weighting some feature maps to contribute more to the result than others.
Subsequent layers will have a problem here because subsequent filtering steps operate across multiple feature maps that will then contribute different amount to the output.

The normalisation operation

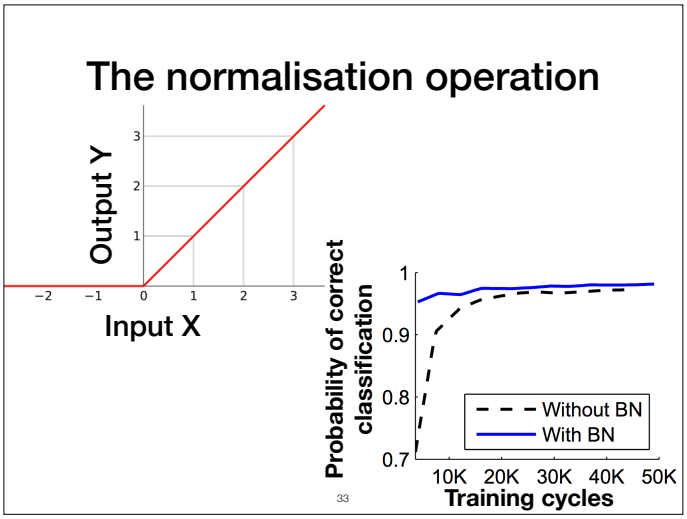


tmp = input-mean(input) then output=tmp/std(input)

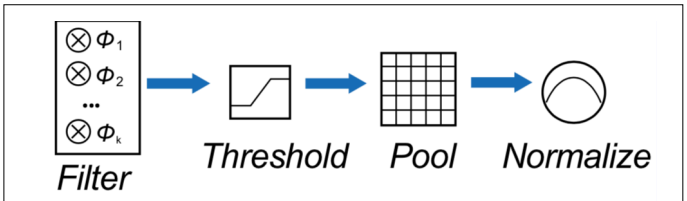
32

So the normalisation operation linearly scales the data to have a mean of zero activation, averaged over all the units in a feature map and all input images.
The first step of normalisation is the subtract the mean response of each feature map from all responses, i.e. to zero-center the data.
The next step is to divide the activation of each layer by its standard deviation.
This makes the normalised data have a mean of zero and a standard deviation of one.

Normalisation is necessary for both theoretical and practical reasons
First, machine learning generally assumes that data reflects measurements of independent and identically-distributed (IID) variables.
Normalisation forces identical distributions.



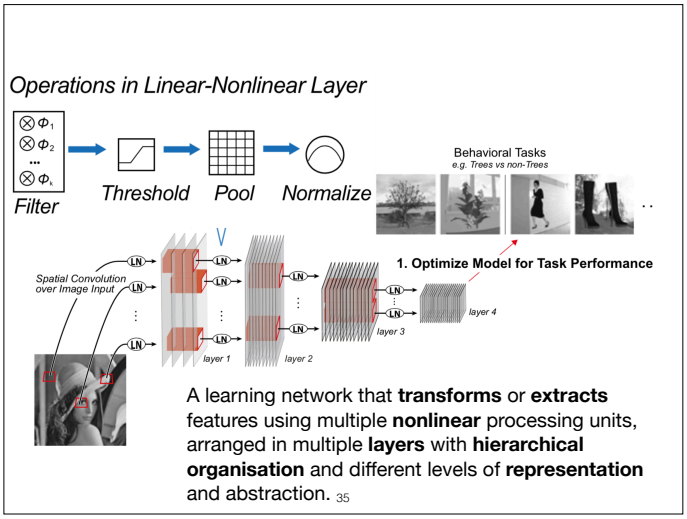
Second, if the threshold activation function depends whether the unit's response is above or below zero, having zero-mean inputs (after normalisation) and zero-mean filters, about half of the units will be active and half inactive. This even split of activation is a very efficient way to store information in a network of limited size.
Third, having the same range for all feature maps and layers means the same threshold (zero) in the threshold function can be used throughout the network.
As a result of these consideration and other technical considerations, network performance increases far more quickly with normalisation, and final classification accuracy by the network is also higher.



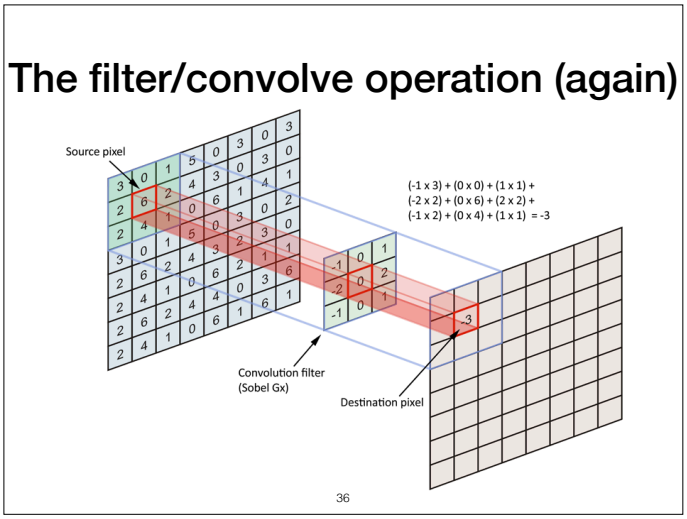
- Filter/convolve: determine how well each group of nearby pixels matches each of a group of filters
- Threshold/rectify: introduce a nonlinearity by setting negative activations of units to zero
- Pool: Downsample the units to improve computational efficiency
- Normalise: Rescale responses of each feature map to have mean zero and standard deviation one, so each feature map contributes similarly to classification

34

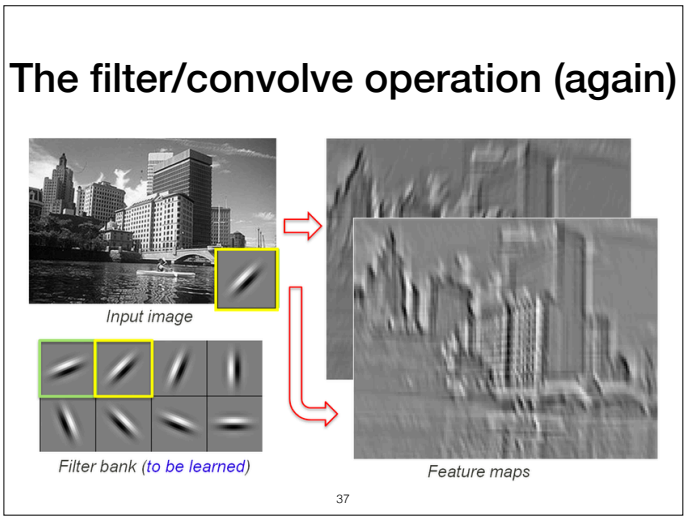
So, we have now done one layer of deep network operations on an image. Let's summarise.



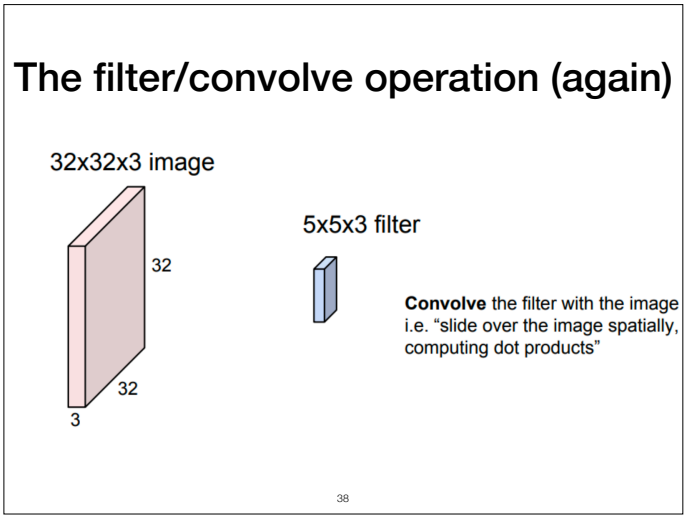
In a deep network, there are several layers performing similar operations and transformations of features, which all follow the same principles. Subsequent layers generally use the same operations in the same way, but the filter/ convolve operation in subsequent layers differs from the what I have just described



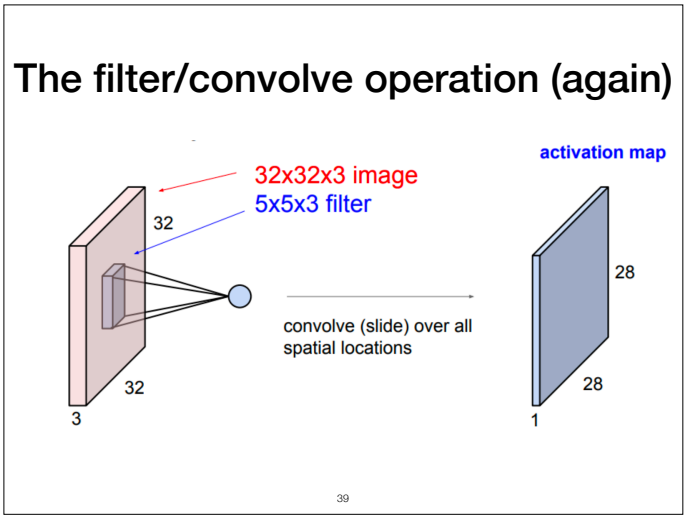
In the first layer, we might be convolving a 2-dimensional input image with a two-dimensional filter to give a 2-dimensional feature map.



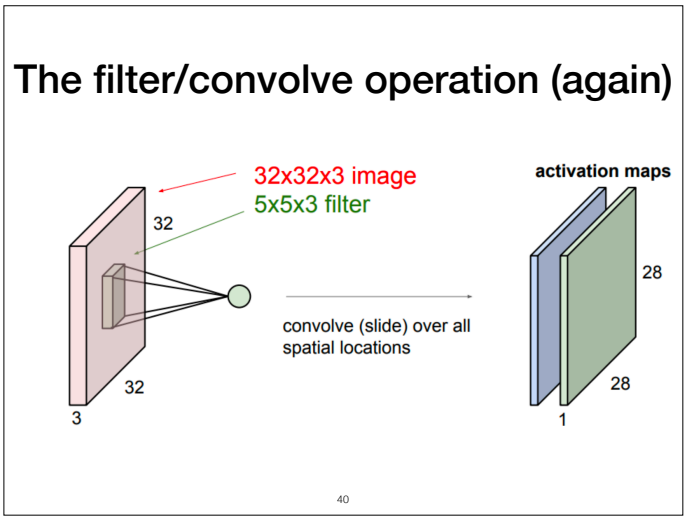
But we do this for several filters, so the next layer is three-dimensional, with the third dimension corresponding to the number of filters used.



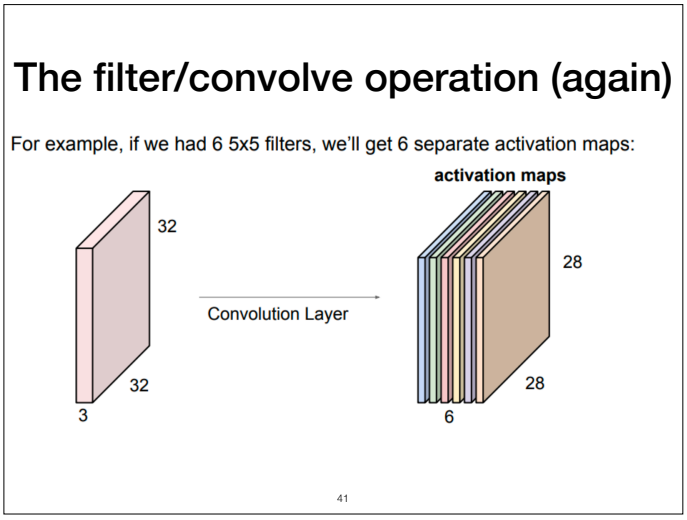
So in subsequent filter operations, the previous layer's activation and the filter are three-dimensional, with the third dimension corresponding to the stack of feature maps in the previous layer's output.
This is even true for some input images: colour images are also treated as three dimensional, with the third dimension being the three colour channels.
So for colour images, the filters used are also three dimensional, with activation then requiring specific colours or different spatial responses in each colour.
The filter will span ALL the feature maps or colour channels in its input.



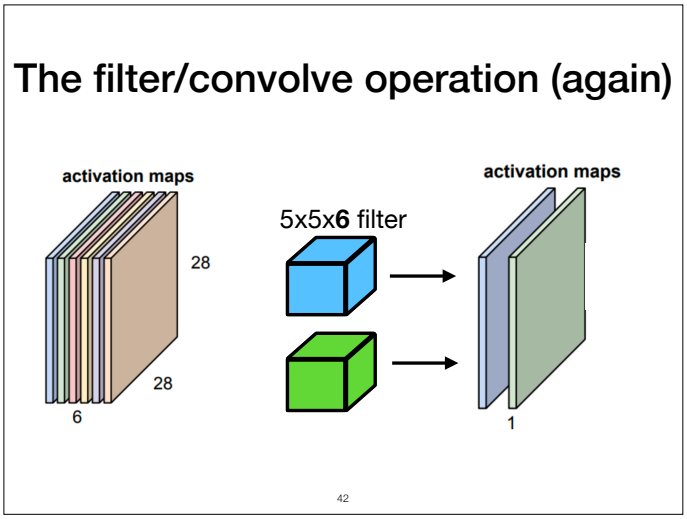
However, the result of a sum of the products of the filter and the corresponding units in the input is still a single number regardless of the size or dimensionality of the inputs to the multiplication. So, even with 3-dimensional inputs and filters, this sum gives the response at a single point in a single feature map of the next layer. Convolved over every position in the input, we then produce a 2-dimensional feature map showing the match between that 3-d filter and the 3-d input at every position.



We repeat this with several 3-dimensional filters acting on the same 3-dimensional input to give several 2-dimensional features maps.

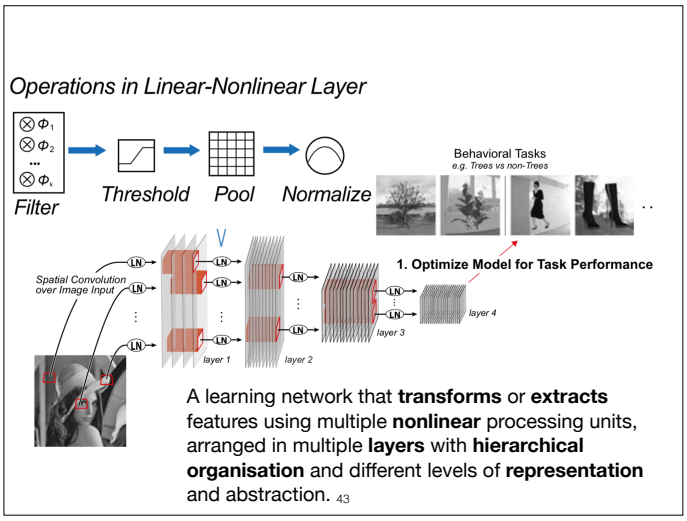


We repeat this with several 3-dimensional filters acting on the same 3-dimensional input to give several 2-dimensional features maps. In this case we used six filters, so we get six output feature maps.

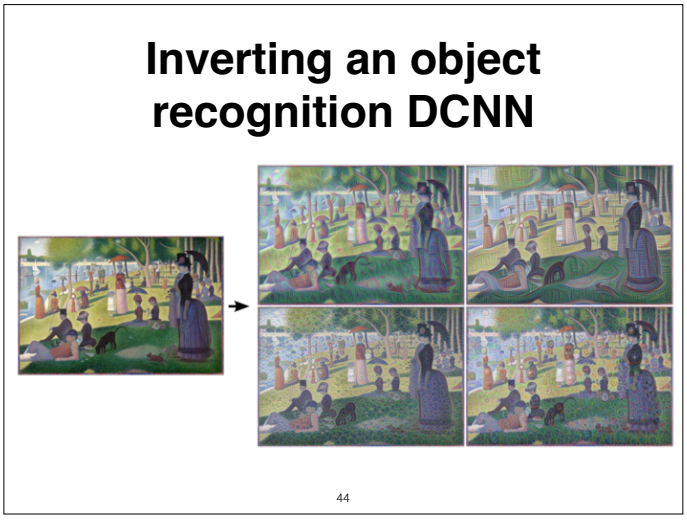


The resulting stack of feature maps is then the input of the next layer. Any filters used in the next layer will operate over ALL of these feature maps, 6 maps in this example. Again, this one filter will produce a single 2-dimensional feature map. Multiple filters will again produce a stack of 2-dimensional feature maps for the next layer.

So we can see the input image as a special case of the inputs to all layers: a black and white input image is like a single feature map, so the filters only need to be a single pixel deep to operate over all feature maps (i.e. one).



As we get higher up the network, these filters get harder to understand in two important ways. First, the filter shape crosses multiple independent feature maps. An edge detector applied to an image is easy enough to conceptualise: We can make images of the 2-dimensional image and the 2-dimensional filter.
But such a 3-dimensional filter crossing multiple feature maps is harder to conceptualise.
Second, the input feature maps become more abstract. It gets very hard to conceptualise what feature is represented.
So both the filter and the feature map become a pattern within a pattern within a pattern. It's very hard to conceptualise these abstract, higher order patterns. Conveniently, we don't need to: the computer does this for us.



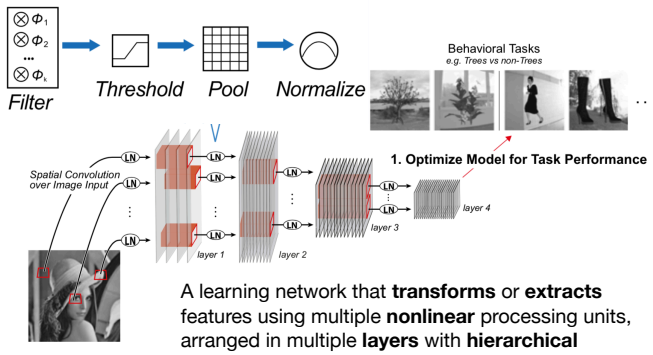
But it is important to understand that deep networks have increasingly complex representations of objects in the images, over several network layers
To get a feel for what features are represented at different levels, we can modify the input image to find a version that produces the strongest possible response at each layer.
After training to classify the objects in natural images, successive layers respond best to increasingly complex relationships between pixels in the input image. These relationships follow the local correlations and patterns found in natural images

Shared weights

- Filters generally have a single set of weights for all positions in the feature map because:
  - If a feature is useful to compute at one position, it is probably also useful at another position.
  - The filter values are weights that need to be learned. Using one filter across all positions greatly reduces the number of weights, improving computational efficiency.
  - The convolution operation is a very fast matrix function. If filters are not fixed, the convolution operation cannot be used.

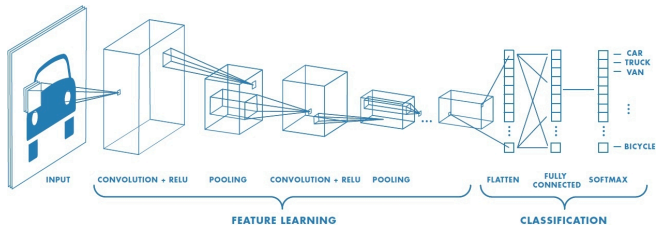
One filter is convolved with the previous feature map stack to give one new feature map.
It would also be possible for the filter to change at different positions in the feature map.
However, in artificial deep networks, generally a single filter is used across all positions, for three very good reasons.
AT END: We will see in our next class only the first of these constraints applies in biological deep networks.

Operations in Linear-Nonlinear Layer



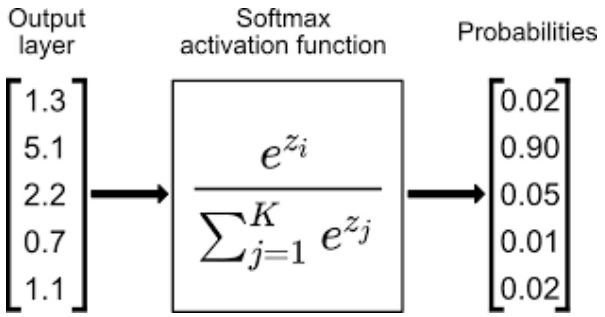
A learning network that **transforms** or **extracts** features using multiple **nonlinear** processing units, arranged in multiple **layers** with **hierarchical organisation** and different levels of **representation** and abstraction. 46

As we get higher in the network, more spatial integration occurs as a result of repeated convolution and pooling, so the spatial dimensions of the feature maps shrink.
At the same time, the number of interesting feature combinations increases, so the feature maps become increasingly narrow, but stacked increasingly high.
In the end, some classification or decision must be made, which is the fundamental goal of the network.



The last stage of the network, after several convolution layers, uses the activity pattern after the final convolutional layer for classification.
To compare this activity pattern with previously-seen patterns, the spatial relationships are first discarded, 'flattening' the last feature map into a line of independent units.
Each of these is connected to units that represent the possible candidate classifications, the labels that describe the input image.
Here we use a fully-connected layer to link every unit to every possible classification with different learned weights.
Essentially the labels see which top-layer pattern they were trained on resembles the current top-layer response pattern most closely.

The softmax operation



48

So, the weights through our network will transform each input image into some ‘score’, reflecting the match between the top layer’s pattern of activation and the pattern of activation by previous examples of each category.
This SCORE must then be converted to a PROBABILITY that this input image falls into each category.
This should take into account not just the score for one category, but also the scores for all other categories: the relative scores determine the probabilities.
This is almost always done with the normalised exponential function, or ‘softmax’.
That is, a constant e raised to the power of the score, divided by the sum of this exponent over all classification scores. As a result, the probabilities sum up to one.
The math is not particularly important to know, but note that, following an exponential function, an output layer score that is only slightly higher than another leads to a probability that is MUCH higher.

Deep learning in artificial neural networks

- Useful for achieving tasks that are difficult to describe formally
  - Difficult for computers, intuitive for humans
- A form of machine learning performing multiple sequential nonlinear feature transformations in hierarchical layers
- Between each feature map layer, a few simple operations
  - Convolution checks each position’s match with a specific filter kernel
  - Thresholding introduces a nonlinearity
  - Pooling downsamples the image, taking the maximum
  - Normalisation rescales responses so each feature map contributes similarly
- Final fully-connected layer links pattern of most abstracted, top-level features to most likely classification
- Softmax determines probability of each classification
  - These probabilities determine the network’s performance in the task

49

Lab Assignment

- Starts by getting familiar with Keras' high level functions and network structure
  - Apply increasingly complex networks to increasingly complex problems
- Then implement these functions yourself to understand how they work on the inside
- Questions available on Friday in Brightspace's 'Assignments' section

- You will be graded in groups of two
  - Do NOT work alone
    - Daan van Rooij can help you find a partner
  - These are the same groups for the 'learning-based' and 'challenge-based' assignment.
- In the 'learning-based' assignments in class, it is most effective if two groups of two (i.e. 4 students) work together
  - More people to share ideas with.
  - Less grading work for our teachers.
  - Each pair should submit a separate document.
- Exercise 3 contains topics for self-study, for the midterm exam
  - Many students find it useful to explore these questions with their group
  - The **deadline for studying these questions is the midterm exam**, which is before the deadline for your lab assignment reports.

Learning-based Lab Assignment

- It is best to work with the same teacher so they can track your progress.
  - Sit in the same section of the same classroom each week.
  - BBG 201, BBG 209, BBG 214
- This is a collaboration
  - You can pair with any classmate.
  - Both group members get the same grade
  - Let your teacher know if your group mate isn't participating
  - If the other pair you work with is not helpful, you don't need to work with them
- For each set of questions:
  - First, think about each answer alone and discuss
  - Agree a common single answer
  - Add this to a shared, combined final answers document (Google docs works well)
    - Show this to your teacher for feedback and grading.
    - Improve if needed, grades can be revised afterwards.
    - Submit **one PDF of the answers per pair**
  - Where your outputs don't match those shown, fix before proceeding
    - Ask for help from your teacher if you can't produce the given results
  - Google Colab lets you work on code together AFTER you have tried it alone. You have plenty of time to do both.