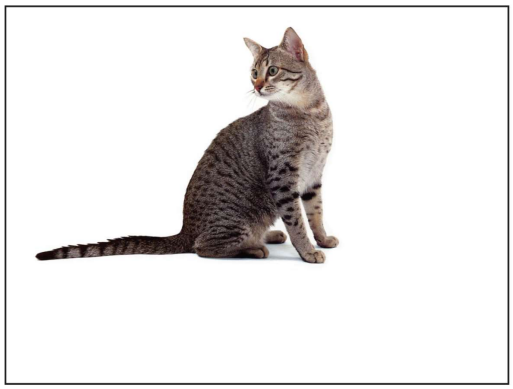
Deep Learning – Equivariance and Invariance

Bernhard Kainz

Invariance and equivariance

- Shift invariance



‘cat’

This image contains a cat

Invariance and equivariance

- Shift invariance

‘cat’

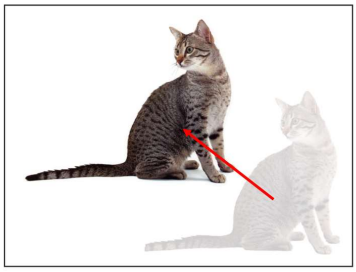
I don’t care where in the image this cat is located. I would like an image discriminator to always give as output ‘cat’

Shift invariance

Input $\mathbf{x}$

Output $f(\mathbf{x}) = 1$

Shifted input $S_v\mathbf{x}$



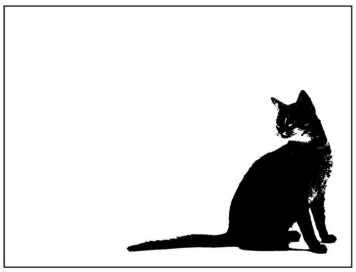
Output $f(S_v\mathbf{x}) = 1$

- **'Cat detector'** $f: \mathbb{R}^d \rightarrow \mathbb{R}$

So I want shift invariance. A network that is insensitive to shifts of the object of interest in an image is called shift invariant. It will not produce a different variant of the output only because the object in the input space underwent a linear translation.
So shift invariance means that the output of the left function and the right function is the same. (in case you didn't notice yet: CNNs are compex function fitting methods. They are neither thinking machines nor intelligent but rather produce an output for a given input)
Let's call the shift operator S with subscript v, where v defines the vector by which I shifted the cat
The shifted input can be thought of some vector space. The shift operator transforms the coordinates in some way where the object is defined.
Shift invariance means that the

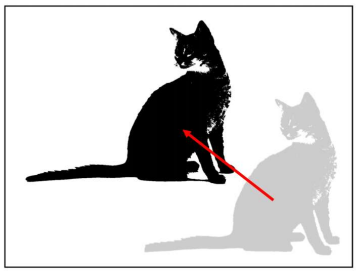
Shift equivariance

Input $\mathbf{x}$



Output $f_i(\mathbf{x}) = \begin{cases} 1 & \text{pixel } i \in \text{cat} \\ 0 & \text{otherwise} \end{cases}$

Shifted input $S_v\mathbf{x}$

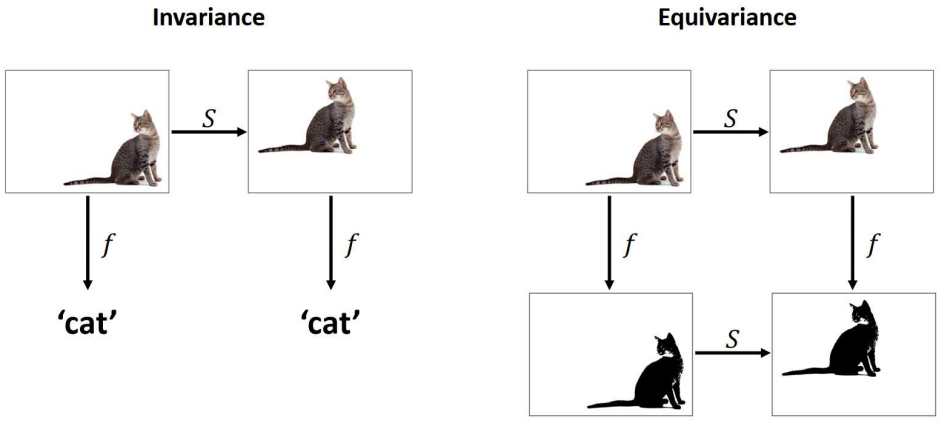


Output $f_i(S_v\mathbf{x}) = \begin{cases} 1 & \text{pixel } i \in \text{cat} \\ 0 & \text{otherwise} \end{cases}$

- **'Cat segmentor' f**: $\mathbb{R}^d \rightarrow \mathbb{R}^d$
- **Shift operator** S_v: $\mathbb{R}^d \rightarrow \mathbb{R}^d$ shifting the image by vector $\boldsymbol{v}$

What is shift equivariance. Well let's say we have now a different model and we want to segment the cat. We want to label every pixel that belongs to the cat as 1 and all the pixels that belong not to the cat as 0.
So in this case the output of the cat detector should shift exactly in the same way as the cat is shifted
So basically shift equivariance means that (click) sfx produces the same result as fsx. In other words f and S commute. F applied to shifted S is the same as S applied to f applied to the output of f.

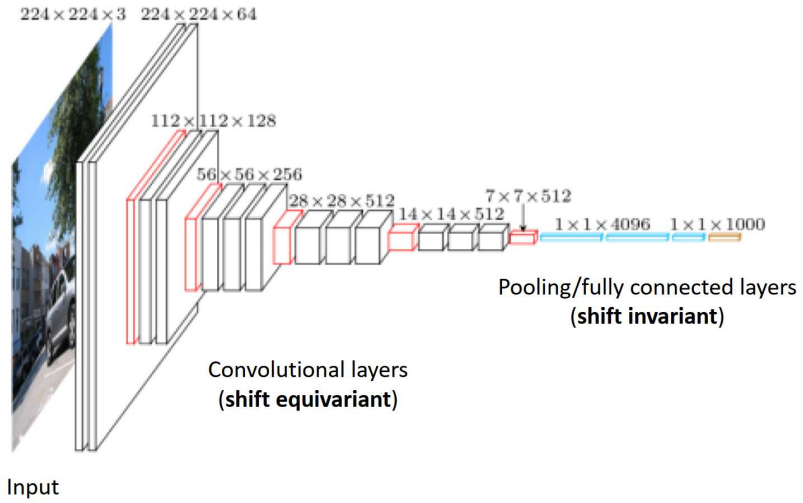
Invariance vs equivariance



Deep Learning – Bernhard Kainz

A way of visualising the difference between invariance and equivariance is this way:
If I talk about invariance I want the ouput to stay constant no matter how I transform the input
If I talk about equivariance I want the output to undergo exactly the same transformation as applied to the input

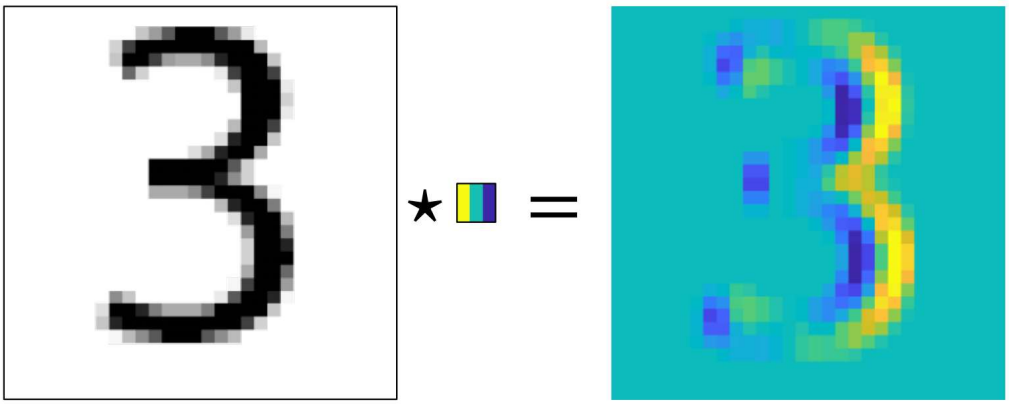
How shift invariance is achieved in CNNs?



So how are these achieved in CNNs.
Basically we have convolutional layers that are supposed to be shift equivariant and pooling layers that are approximately shift invariant.

So why is that

Equivariance in CNNs

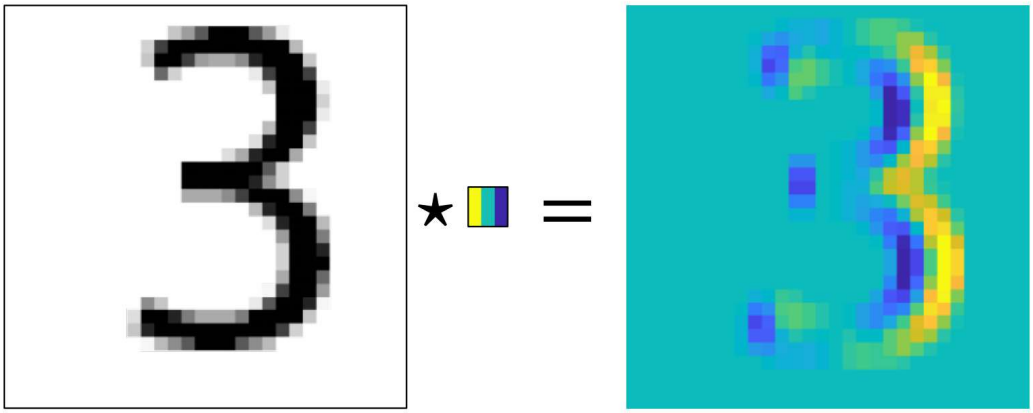


Output of convolutional layer (shift equivariant)

Deep Learning – Bernhard Kainz

In theory, applying a filter kernel to an image should not change the filter response. This can be proven in frequency space and through the Fourier transform where we can find that convolutions in the image or time domain (remember back the animations about two convoluted functions) is equivalent to a multiplication in frequency space.
So in theory convolutions should be **shift equivariant**

Equivariance in CNNs

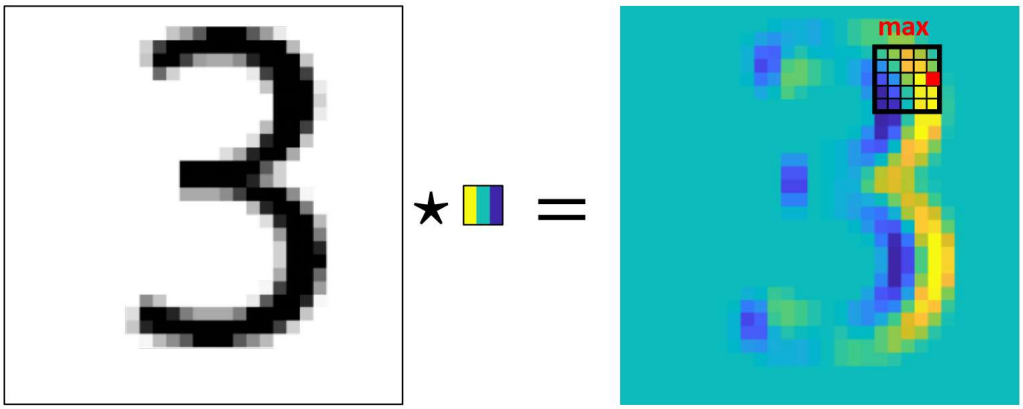


Output of convolutional layer (shift equivariant)

Deep Learning – Bernhard Kainz

So if I shift the image a little bit, the filters will produce the same response at the location of the input object

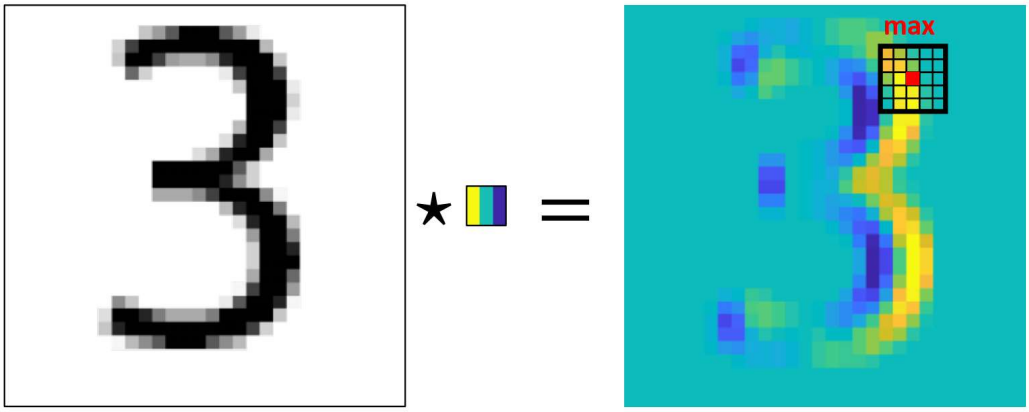
Approximate invariance in CNNs with pooling



Output of convolutional layer+max pooling (~shift invariant)

Now, pooling builds up shift invariance. Approximately
If I shift the image a little bit under this pooling kernel, the maximum will still be the same.
It is not completely bullet proof but at least locally it will still find the same maximum in this small window.

Approximate invariance in CNNs with pooling

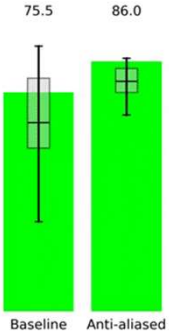


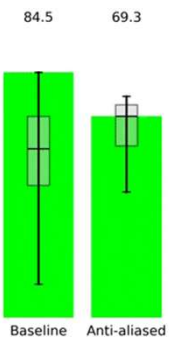
Output of convolutional layer+max pooling (~shift invariant)

Not the full story…

Nyquist sampling theorem = sample at least twice as fast to keep all information

- But striding ignores the Nyquist sampling theorem and aliases





BK1

R. Zhang.
Making Convolutional Networks Shift-Invariant Again.
In ICML, 2019.

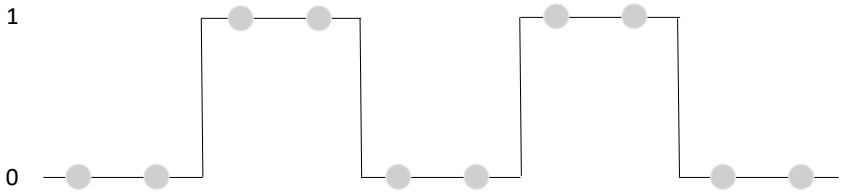
These things are in general true but they are not the full story.

Slide 103

BK1 Bernhard Kainz, 15/09/2020

Simple example

- Max-pooling breaks shift equivariance

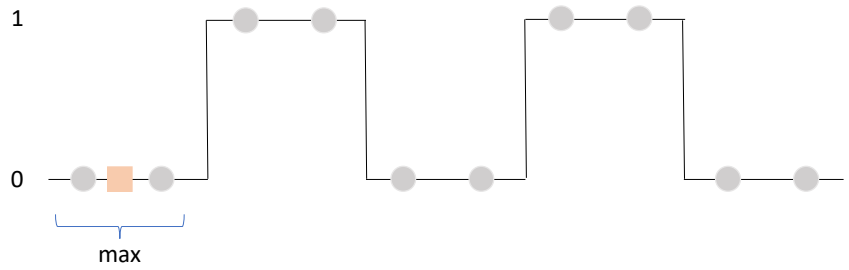


https://www.youtube.com/watch?v=eZa56DqXTHg

Simple toy signal. It goes from 0 to 1 and back

Simple example

- Max-pooling breaks shift equivariance

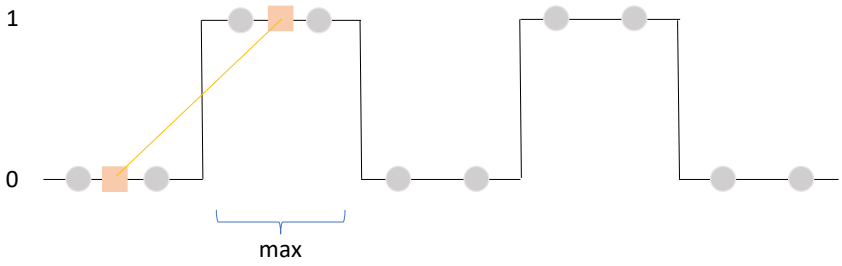


https://www.youtube.com/watch?v=eZa56DqXTHg

Let's max pool it together
0 and 0 is 0

Simple example

- Max-pooling breaks shift equivariance

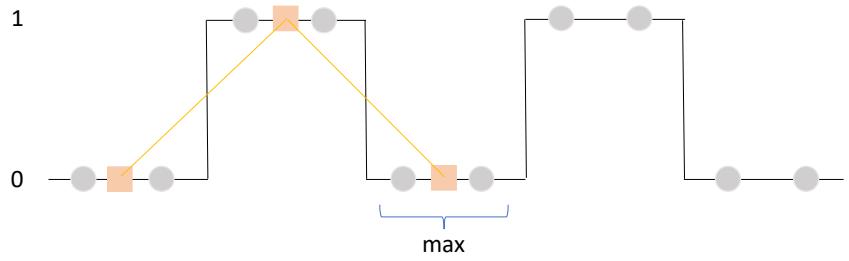


https://www.youtube.com/watch?v=eZa56DqXTHg

Let's max pool it together
0 and 0 is 0

Simple example

- Max-pooling breaks shift equivariance

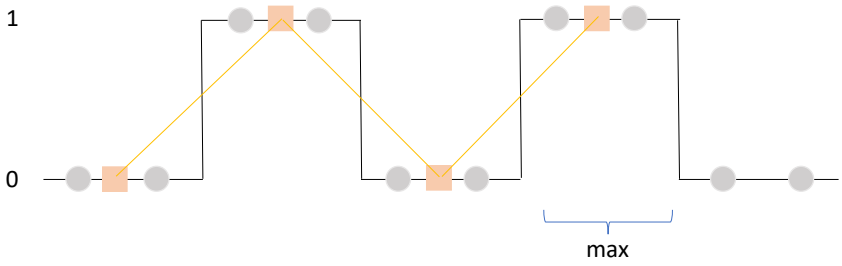


https://www.youtube.com/watch?v=eZa56DqXTHg

Let's max pool it together
0 and 0 is 0

Simple example

- Max-pooling breaks shift equivariance

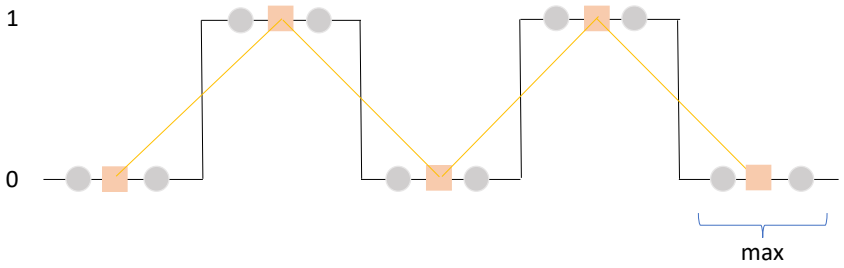


https://www.youtube.com/watch?v=eZa56DqXTHg

Let's max pool it together
0 and 0 is 0

Simple example

- Max-pooling breaks shift equivariance

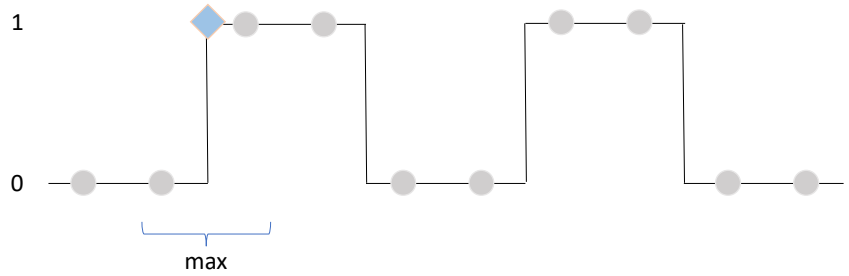


https://www.youtube.com/watch?v=eZa56DqXTHg

Let's max pool it together
0 and 0 is 0

Simple example

- Max-pooling breaks shift equivariance

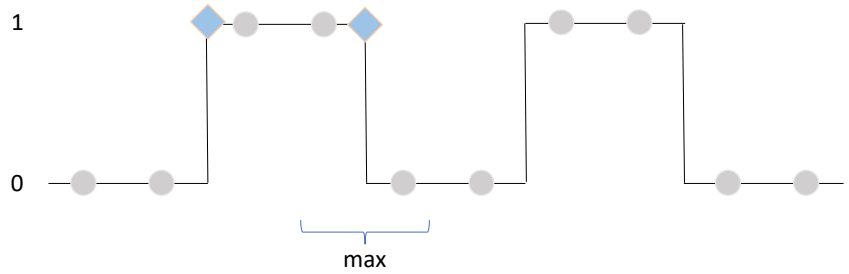


https://www.youtube.com/watch?v=eZa56DqXTHg

Now if we shift the signal by one index we get the max of 0 and 1

Simple example

- Max-pooling breaks shift equivariance

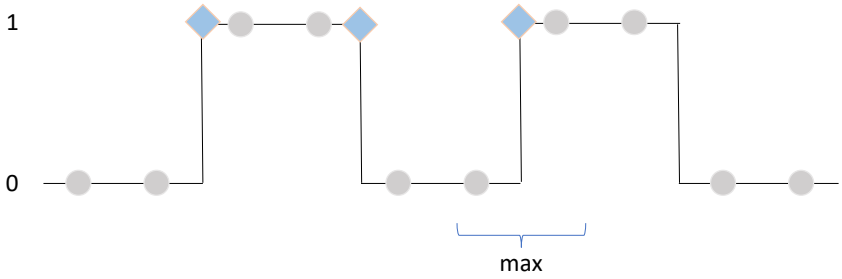


https://www.youtube.com/watch?v=eZa56DqXTHg

Now if we shift the signal by one index we get the max of 0 and 1

Simple example

- Max-pooling breaks shift equivariance

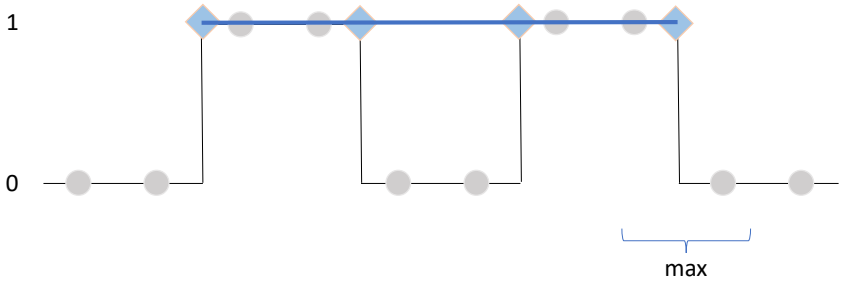


https://www.youtube.com/watch?v=eZa56DqXTHg

Now if we shift the signal by one index we get the max of 0 and 1

Simple example

- Max-pooling breaks shift equivariance



https://www.youtube.com/watch?v=eZa56DqXTHg

Now if we shift the signal by one index we get the max of 0 and 1

Simple example

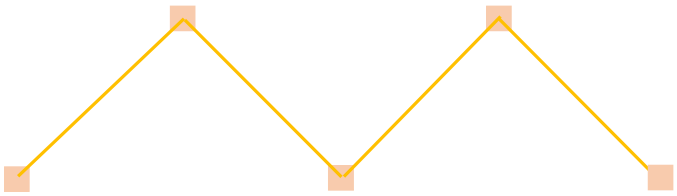
- Max-pooling breaks shift equivariance

https://www.youtube.com/watch?v=eZa56DqXTHg

Now if we shift the signal by one index we get the max of 0 and 1

Simple example

- Max-pooling breaks shift equivariance



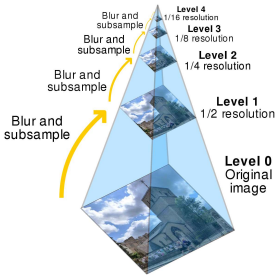
https://www.youtube.com/watch?v=eZa56DqXTHg

Now you would probably agree that these two signals look very different.
What we have done here is that we have broken shift equivariance.

Simple example

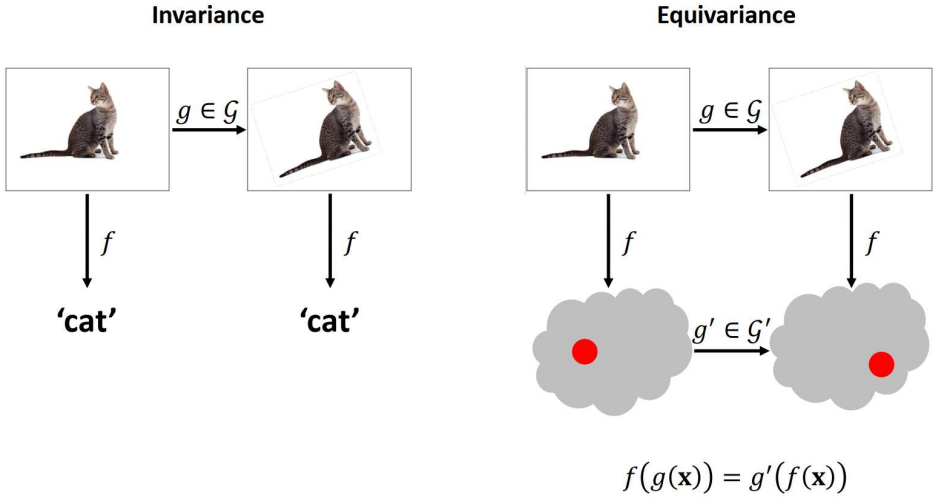
- Max-pooling breaks shift equivariance
- Partial solution: use what you learned about anti-aliasing in Computer Vision: blur and then down sample

R. Zhang.
Making Convolutional Networks Shift-Invariant Again.
In ICML, 2019.



https://www.youtube.com/watch?v=eZa56DqXTHg

Beyond shifts: group equivariance

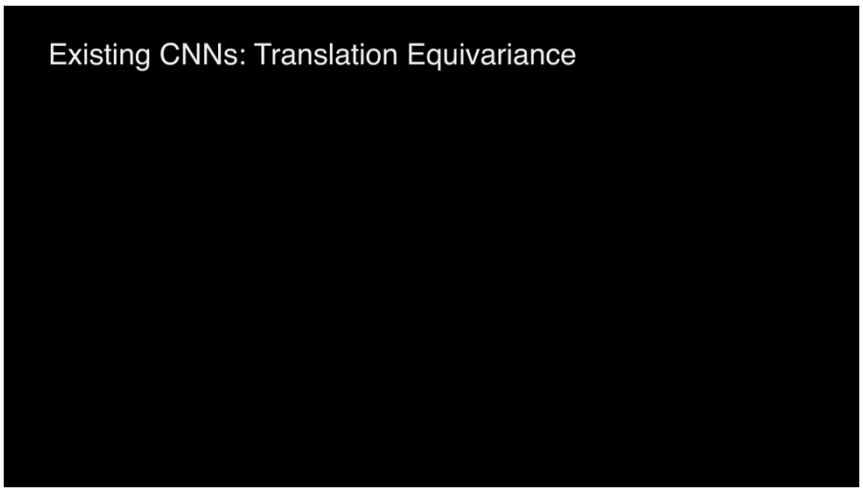


$$f(g(\mathbf{x})) = g'(f(\mathbf{x}))$$

Deep Learning – Bernhard Kainz

We can extend this beyond translational shift and apply this to any group operation. Groups are simply collections of operations that have the property of closure. These are any operations where if I combine two elements of that group I get another element of that group. Simple example: rotations: I can first rotate a little bit to the left and then another little bit and if I combine these I get a consolidated rotation matrix that will do exactly the same as the two previous ones.

Rotation invariant CNNs



Daniel Worrall et al.: Harmonic Networks: Deep Translation and Rotation Equivariance

https://www.youtube.com/watch?v=goWAFBYOtoU

Achieving group invariance like for example rotation invariant CNNs is reasonable challenging. The paper in this work applied spherical harmonics to achieve this.
If you look at the features maps when rotated the filters in this classification network react differently to different rotations. With harmonics the feature maps stays more or less constant.

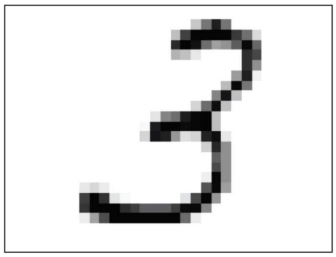
Rotation invariant CNNs

Daniel Worrall et al.: Harmonic Networks: Deep Translation and Rotation Equivariance
https://www.youtube.com/watch?v=goWAFBYOtoU

Achieving group invariance like for example rotation invariant CNNs is reasonable challenging. The paper in this work applied spherical harmonics to achieve this.
If you look at the features maps when rotated the filters in this classification network react differently to different rotations. With harmonics the feature maps stays more or less constant.

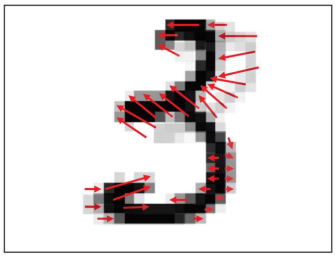
Approximate deformation invariance

Input $\mathbf{x}$



Output $f(\mathbf{x}) = 1$

Shifted input $S_v\mathbf{x}$



Output $f(D_\tau\mathbf{x}) = 1$

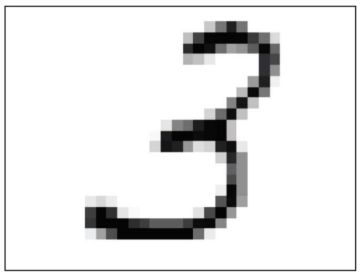
- **'Digit 3 detector'** $f: \mathbb{R}^d \to \mathbb{R}$
- **Warp operator** $D_\tau: \mathbb{R}^d \to \mathbb{R}^d$ warping the image by field $\boldsymbol{\tau}$

Deformation invariance: $f(\mathbf{x}) \approx f(D_\tau\mathbf{x})$

This can also be generalised to operations that are not necessarily described by groups and in particular we can talk about deformations
We can think of warping operations like small local pertubations – like variants of smooth transformations of a canonical 3 like in this image/
So we can write this like a warping operator describing a smooth deformation field that is applied to pixels and moves them a little bit
We can think of these deformations as small local shifts, so CNNs are so successful for example classifying these handwritten digits because they are approximately invariant to these shifts.

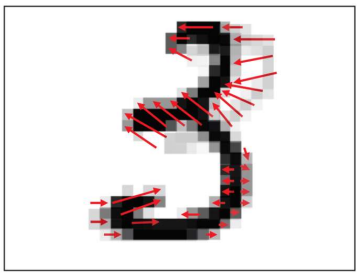
Approximate deformation invariance

Input $\mathbf{x}$



Output $f(\mathbf{x}) = 1$

Shifted input $S_v\mathbf{x}$



Output $f(D_{\boldsymbol{\tau}}\mathbf{x}) = 1$

- **'Digit 3 detector'** $f: \mathbb{R}^d \to \mathbb{R}$
- **Warp operator** $D_{\boldsymbol{\tau}}: \mathbb{R}^d \to \mathbb{R}^d$ warping the image by field $\boldsymbol{\tau}$

$$\|f(\mathbf{x}) - f(D_{\boldsymbol{\tau}}\mathbf{x})\| \approx \|\nabla \boldsymbol{\tau}\|$$

Deep Learning – Bernhard Kainz

Success now mainly depends on how this vector field tau is different from constant shift.