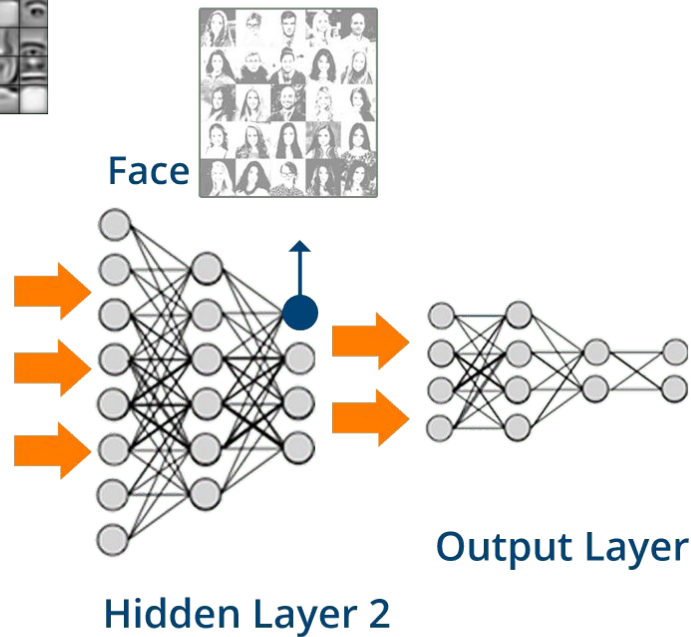
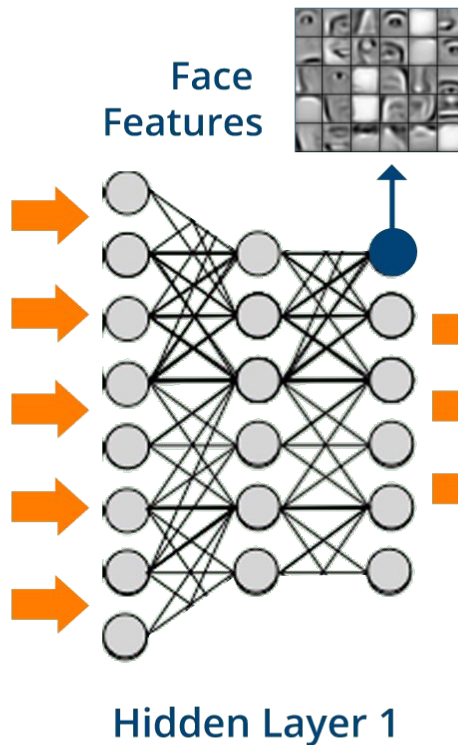
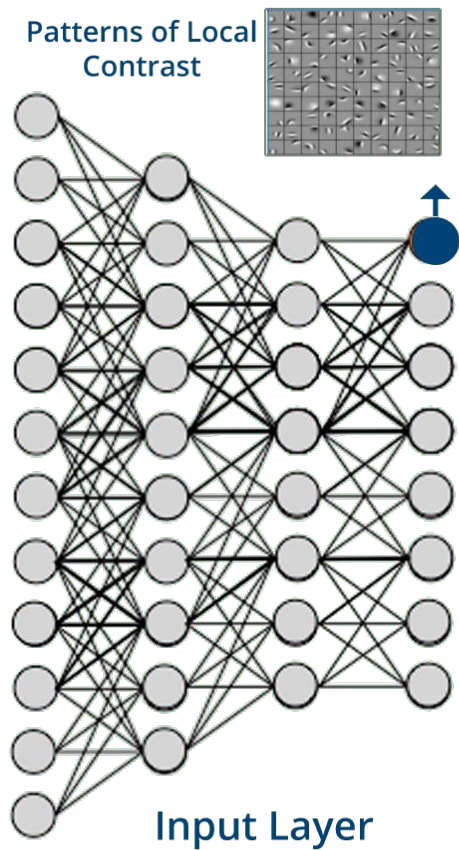


# Review: Machine Learning Intuitions


Things you should know by heart <3

# A Review of Machine Learning Intuitions

- Activation Functions
- Epoch
- Underfit vs Overfit
- Overfitting Analogy and Dropouts
- Learning Rate
- Optimizers
- Batch Size
- Steps per Epoch
- Convolution, Dilation, Stride (subsample) , Filters (kernel)
- Layer visualization



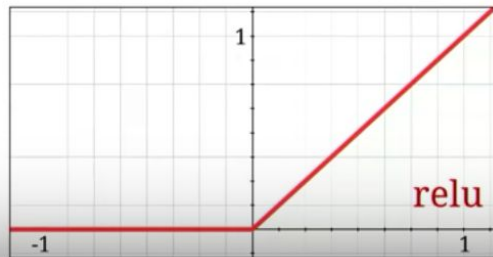
# Activation Functions


$$Y = \text{relu}\left(\sum_i W_i X_i + b\right)$$

activation

weights

bias



Hidden layers

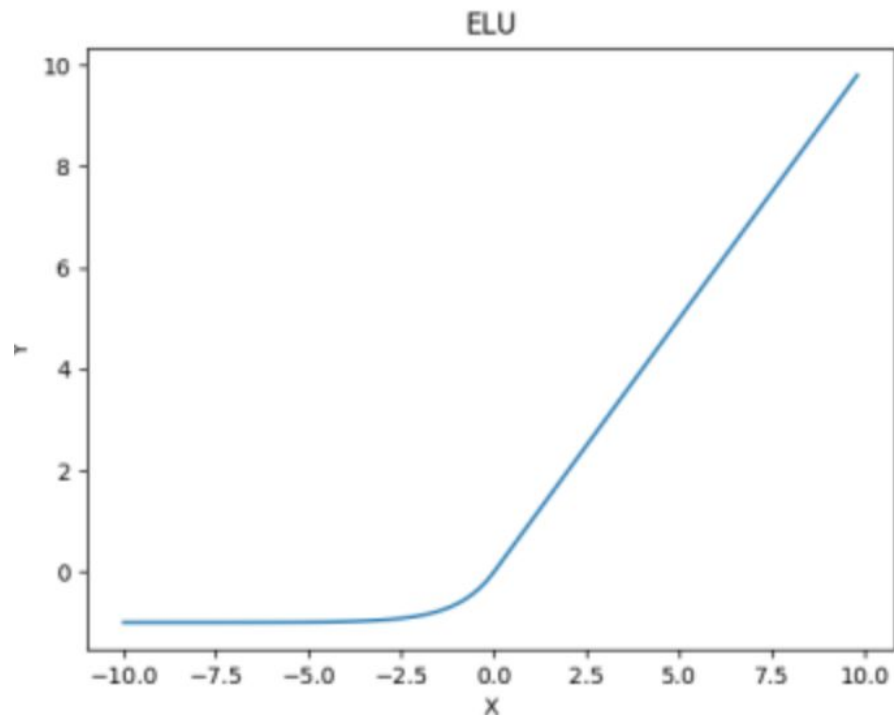
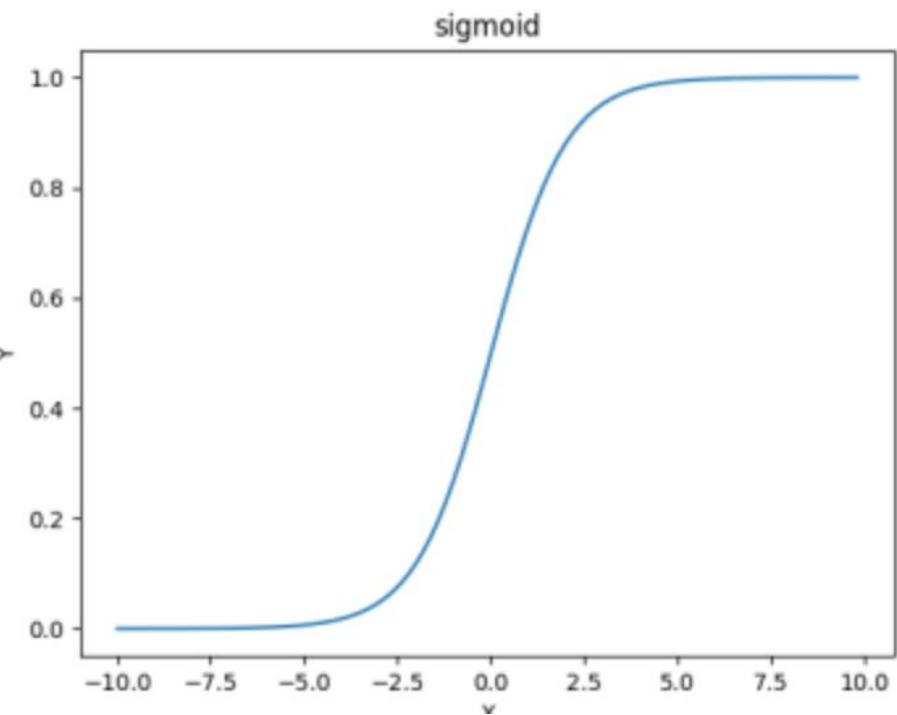
TensorFlow and deep reinforcement learning, without a PhD (Google I/O '18)

<https://www.youtube.com/watch?v=t1A3NTttvBA&t=143s>










# Activation Functions

The activation function of a node defines the output of that node given an input or set of inputs.

# Activation Functions



# Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

# Activation Functions - curated links

## **Understanding Activation functions**

<https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>

## **Comparison of non-linear activation functions**

<https://arxiv.org/pdf/1804.02763.pdf>

## **Exponential Linear Units are the New Cool**

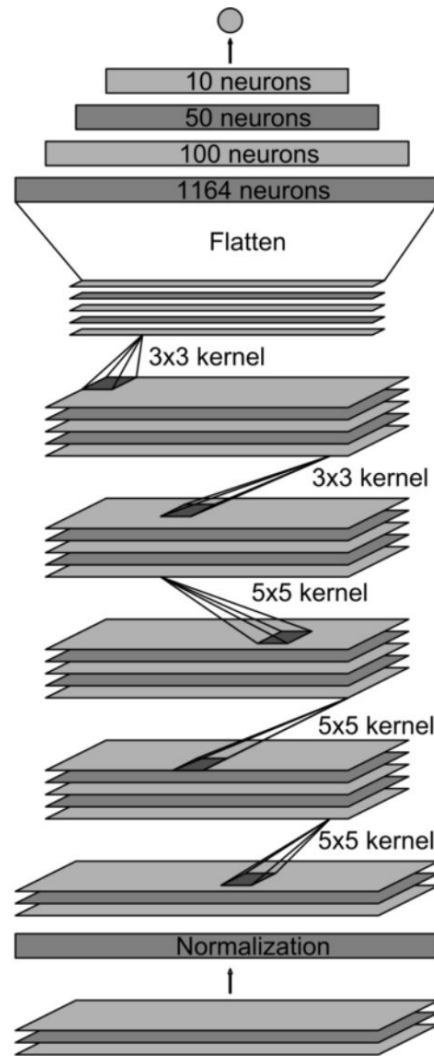
<http://saikatbasak.in/sigmoid-vs-relu-vs-elu/>



# NETWORK ARCHITECTURE

- 9 layers,
- 1 normalization layer
- 5 convolutional layers
- 3 fully connected layers.

From:  
<https://devblogs.nvidia.com/deep-learning-self-driving-cars/>



Output: vehicle con...

Fully-connected lay

Fully-connected lay

Fully-connected lay

Convolutional

feature map

64@1x18

Convolutional

feature map

64@3x20

Convolutional

feature map

48@5x22

Convolutional

feature map

36@14x47

Convolutional

feature map

24@31x98

Normalized

input planes

3@66x200

Input planes

3@66x200

```
def build_modified_nvidia_model():  
    model = Sequential()  
    model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))  
    model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))  
    model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))  
    model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))  
    model.add(Conv2D(64, 3, 3, activation='elu'))  
    model.add(Conv2D(64, 3, 3, activation='elu'))  
    model.add(Dropout(0.5))  
    model.add(Flatten())  
    model.add(Dense(100, activation='elu'))  
    model.add(Dense(50, activation='elu'))  
    model.add(Dense(10, activation='elu'))  
    model.add(Dense(1))  
    model.compile(optimizer = OPTIMIZER_TYPE, loss = LOSS_TYPE)  
    return model
```

# Epoch

- An epoch is one complete pass of all the data in our data set.
- So 30 epochs means we pass the dataset 30 times.
- We must be aware of this since a large epoch size might overfit our network while a smaller size will underfit it.

## Overfit vs Underfit (1/2)

- Overfit means that it might work very well at our training data set but won't generalize to new data.
- Underfit means that the model works poorly both with our training data and new data.

## Overfitting Analogy

*Overfit: the guy who has a perfect grade in school, in all subjects, but outside school knows nothing in real world.*

- Arnaldo Gunzi (<https://medium.com/@arnaldogunzi>)

## Overfitting Analogy

*Or someone who has a PhD in nuclear advanced theoretical gravitational quantum physics, but works as a waiter in a restaurant, because his knowledge is so specific it has no real world application.*

- Arnaldo Gunzi
- <https://medium.com/@arnaldogunzi>

# Learning Rate

- The intuition behind learning rate is how quickly a network abandons old beliefs for new ones.
- We want to find a learning rate that is low enough that the network quickly becomes useful, but high enough that you don't have to spend years training it.

## Learning Rate Example (1/3)

*If a child sees 10 examples of cats and all of them have orange fur, it will think that cats have orange fur and will look for orange fur when trying to identify a cat.*

*Now it sees a black a cat and her parents tell her it's a cat (supervised learning).*

- Conner Davis

<https://www.quora.com/What-is-the-learning-rate-in-neural-networks>



## Learning Rate Example (2/3 )

*With a large “learning rate”, it will quickly realize that “orange fur” is not the most important feature of cats.*

*With a small learning rate, it will think that this black cat is an outlier and cats are still orange.*

- Conner Davis

<https://www.quora.com/What-is-the-learning-rate-in-neural-networks>

## Learning Rate Example (3/3 )

*The point is that a higher learning rate means the network changes its mind more quickly. That can be good in the case above, but can also be bad. If the learning rate is too high, it might start to think that all cats are black even though it has seen more orange cats than black ones.*

- Conner Davis

<https://www.quora.com/What-is-the-learning-rate-in-neural-networks>

# Learning Rate

- If the learning rate is too high then it becomes like a wishy-washy person who always misses the point every time you explain things differently or provide a new example of an idea.

## Learning Rate

- If the learning rate is too low then it's like a stupid person who can eventually understand something but takes so long it's annoying.

## Learning Rate

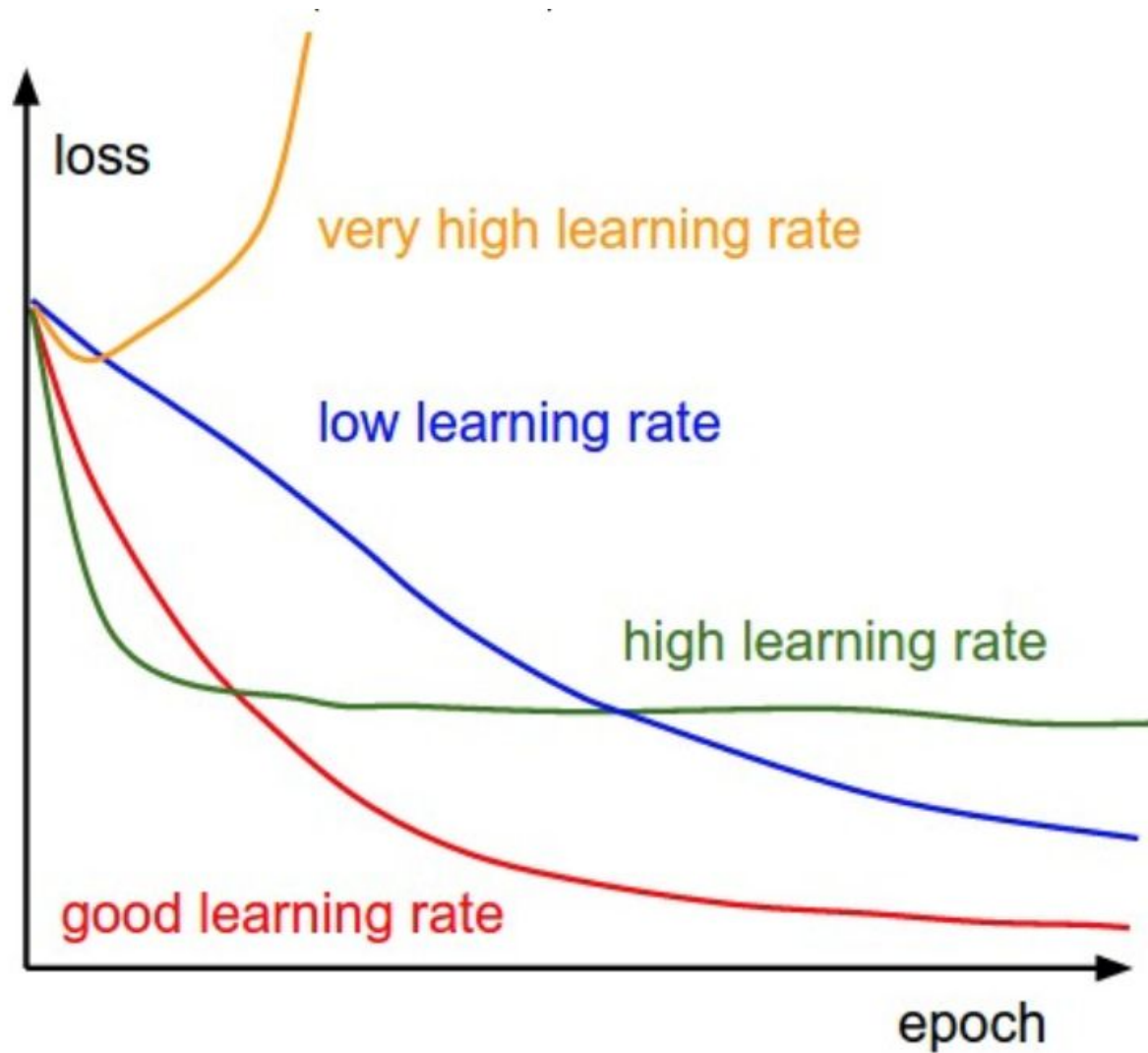
When someone has explained something to you 8 times and you still don't get it and you hope they forgive how stupid you are



## Learning Rate

- We want the network to have a learning rate (*or in our case an initial adaptive learning rate*) such that it behaves like a reasonably intelligent person.

# Learning Rate



# Optimizers

- Algorithms used to minimize the *objective function*
- Basically, the algorithm we use to update the weights during training

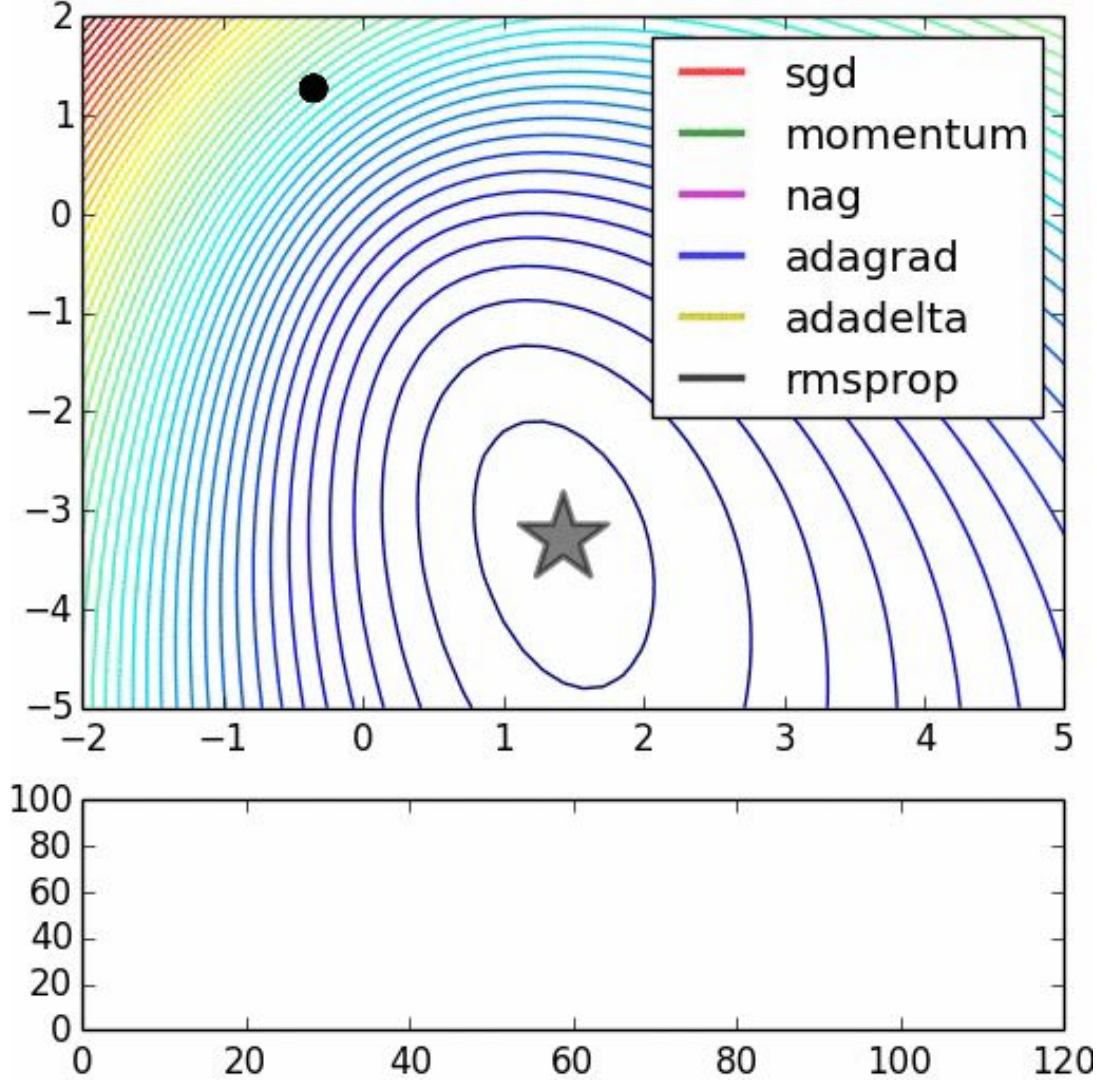
*\*Objective Function - The function that calculates the loss/error like MSE*



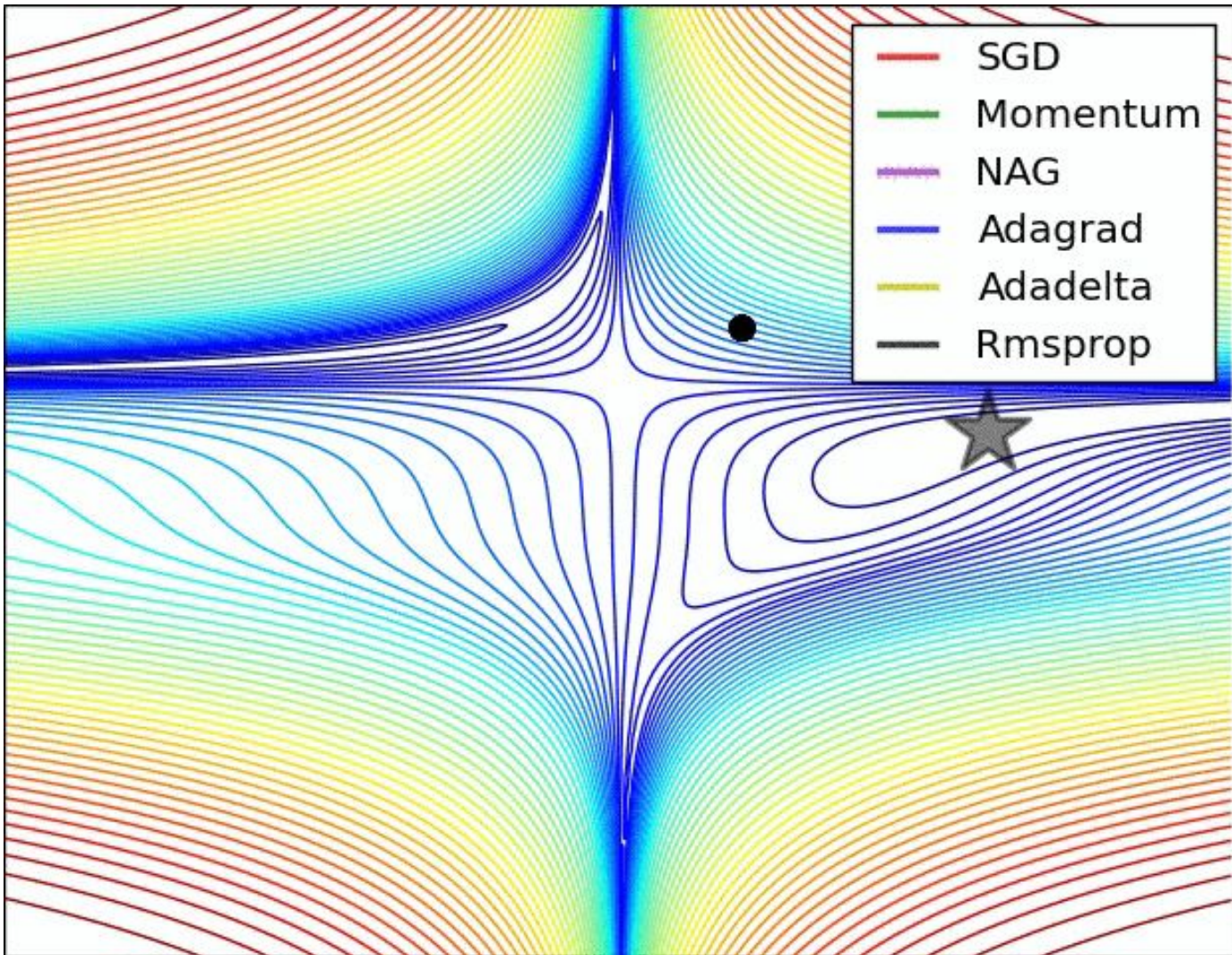
# Optimizers

- Adam (popular, works well in practice), but there are many others you can check out.
- <https://keras.io/optimizers/#usage-of-optimizers>
- <https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>

# Optimizers

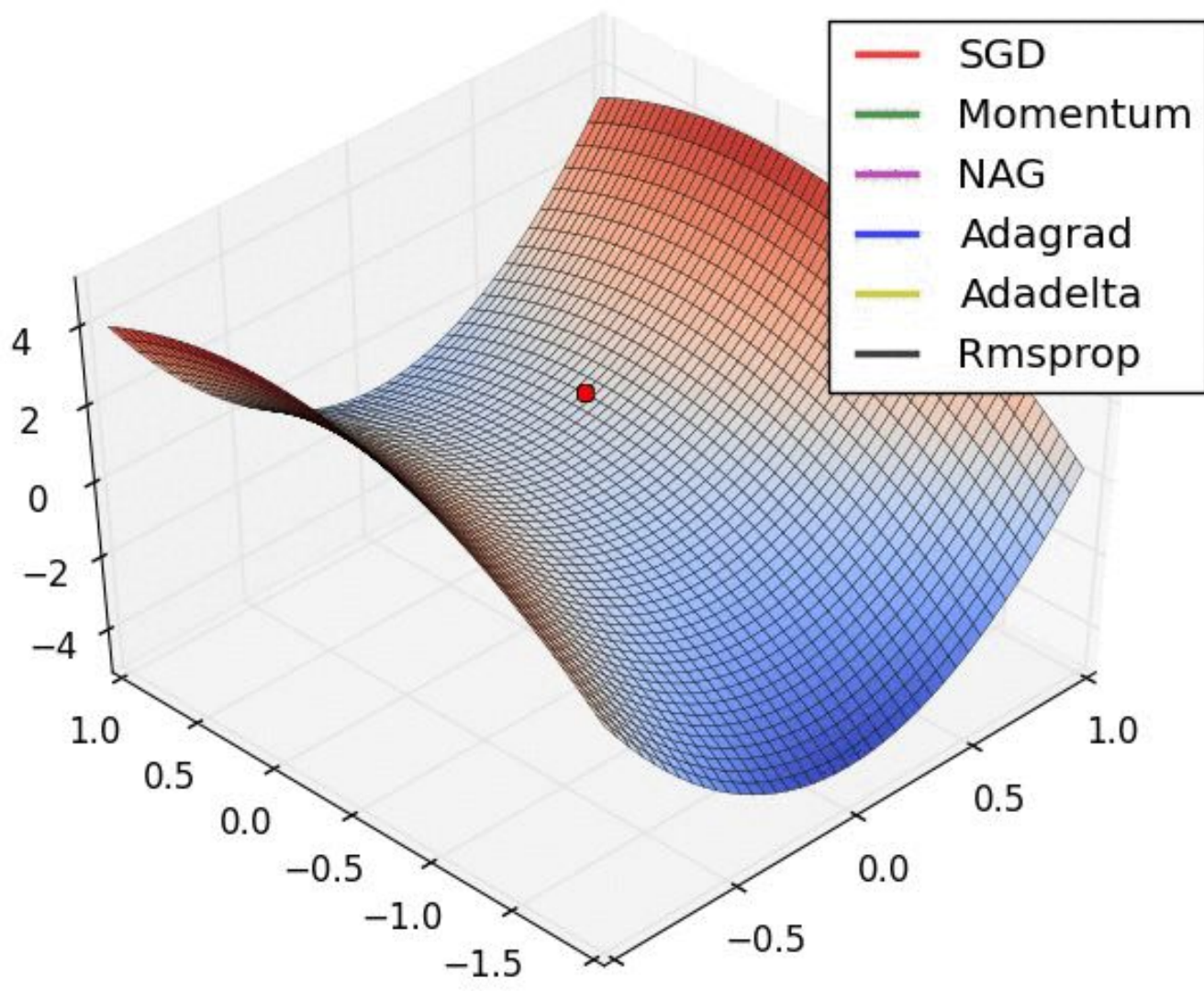


# Optimizers





# Optimizers

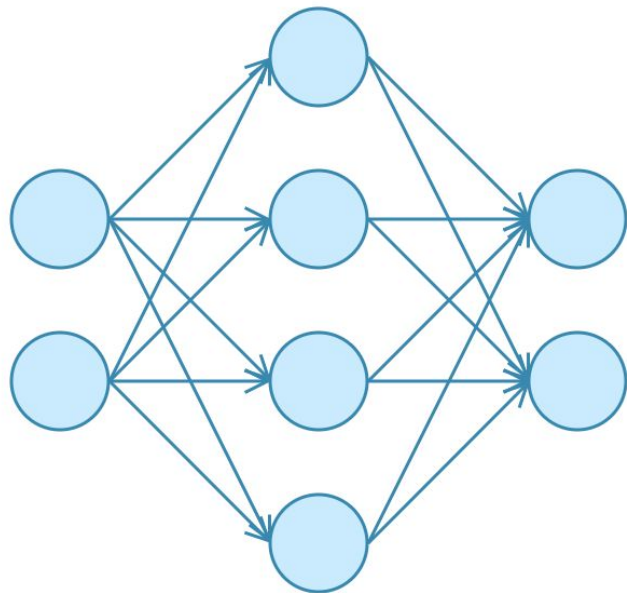


# Dropouts

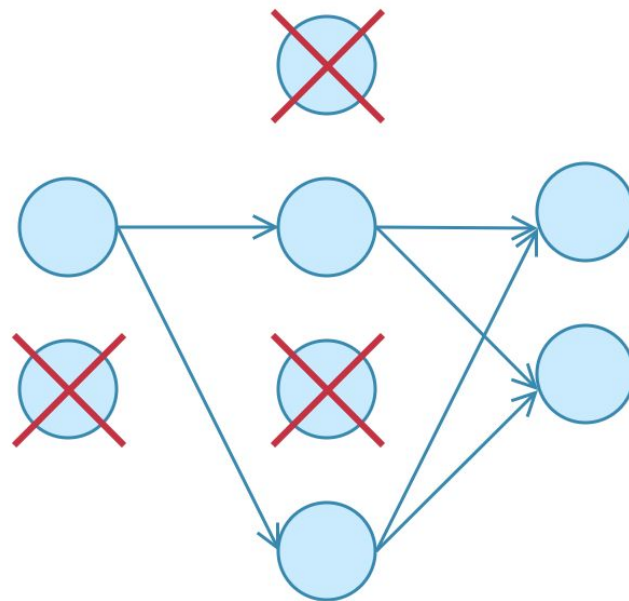
- To prevent overfitting!
- Almost all state-of-the-art deep networks now incorporate dropout.
- During training time, at each iteration, neurons are temporarily or disabled with probability  $p$  (dropout rate) randomly.

# Dropouts

[towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2](https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2)



No Dropout



With Dropout

## Dropouts

*You were the only person at your company who knows about ML. Every morning you tossed a coin to decide whether you will go to work or not. Some days you might not be at work but ML tasks still need to get done, so they can't rely only at you. Sometimes your coworkers also toss a coin to decide whether they will go or not.*

<https://towardsdatascience.com/applied-deep-learning-part-4-revolutional-neural-networks-584bc134c1e2>

## Dropouts

*Your coworkers will need to learn about ML and this expertise needs to be spread out between various people. The workers need to cooperate with several other employees, not with a fixed set of people. This makes the company much more resilient overall, increasing the quality and skill set of the employees.*

<https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>



## Batch Size

- Computing the gradient over the entire dataset is expensive and slow. This is why we use batches.

## Batch Size

- *This is usually 32- 512 data points.*
- *In terms of computational power, while the training process takes many many more iterations, you end up getting there for less cost than the full batch mode, "typically."*

<https://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network/236393>

## Batch Size

- *Optimizing the exact batch size batch you left to trial and error.*
- *Batch size, learning rate - linked.*
- *Batch size is too small then the gradients become more unstable...need to reduce the learning rate.*

<https://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network/236393>

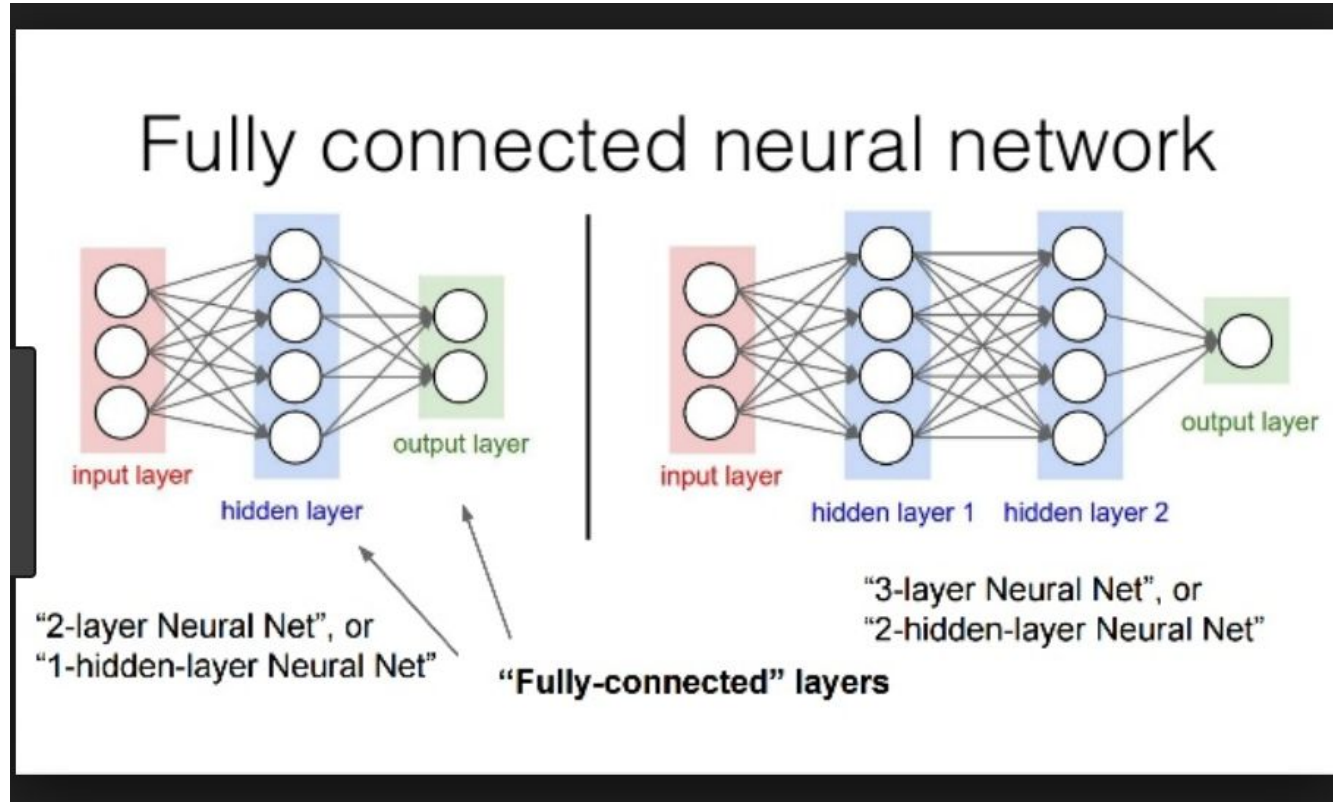
## Batch Size

- *observed in practice - when using a larger batch significant degradation in model's ability to generalize.*
- <https://stats.stackexchange.com/questions/164876/tradeoff-batch-size-vs-number-of-iterations-to-train-a-neural-network/236393>

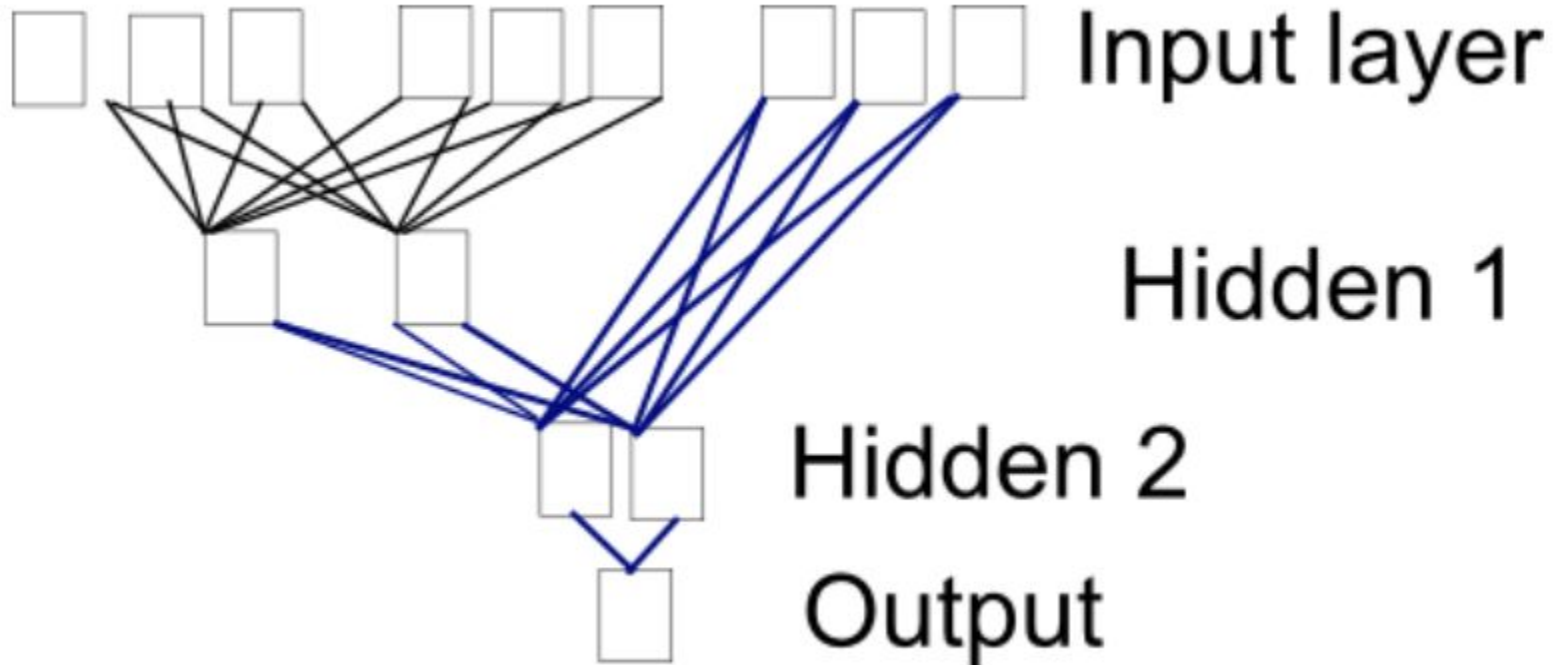
## Steps Per Epoch

- The number of steps (batches of samples) before declaring your epoch finished.
- The number of training images over batch size
- Why? Because you should train your entire data on every epoch.
- Similarly, **validation steps per epoch** should be number of validation images over batch

# Fully-connected Layers (DENSE)



## NOT Fully-connected Layers



# Convolution

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

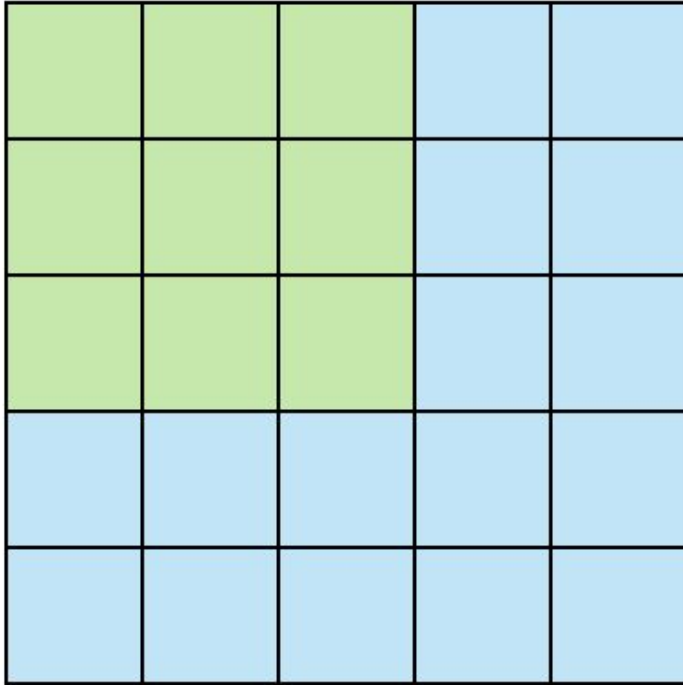
Convolved  
Feature

*Typical sliding  
window!!!*

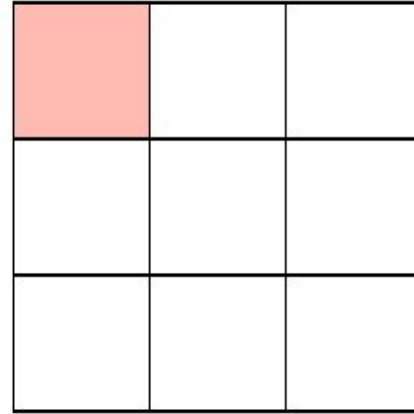


# Stride (Subsample)

<https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>



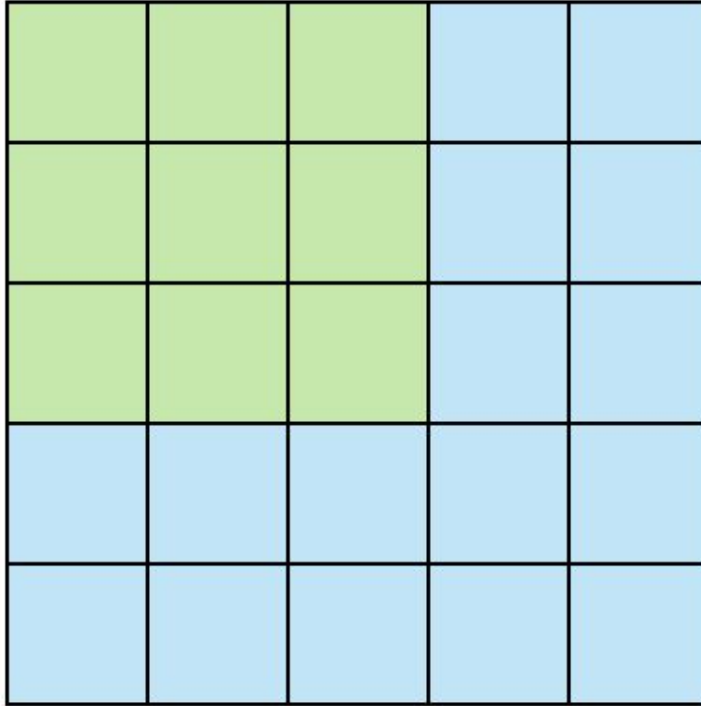
Stride 1



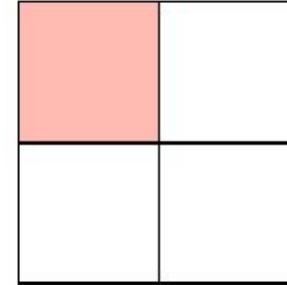
Feature Map

# Stride (Subsample)

<https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>



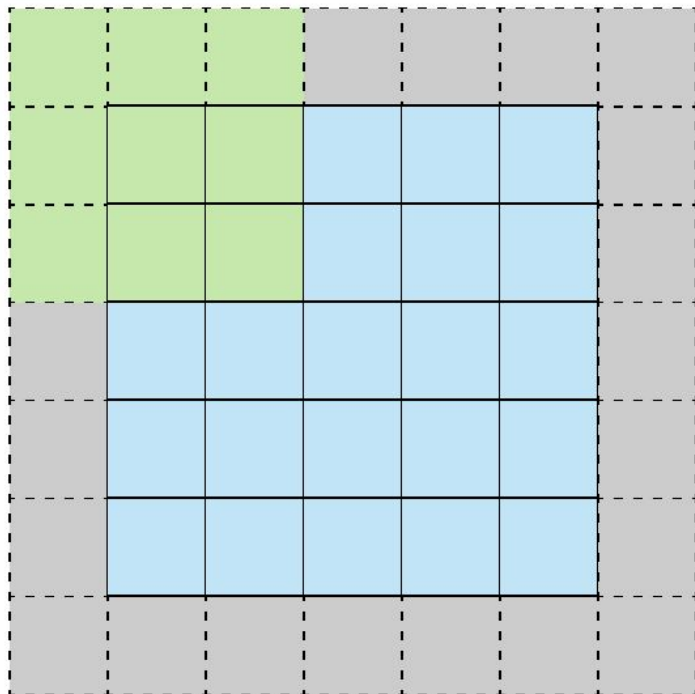
Stride 2



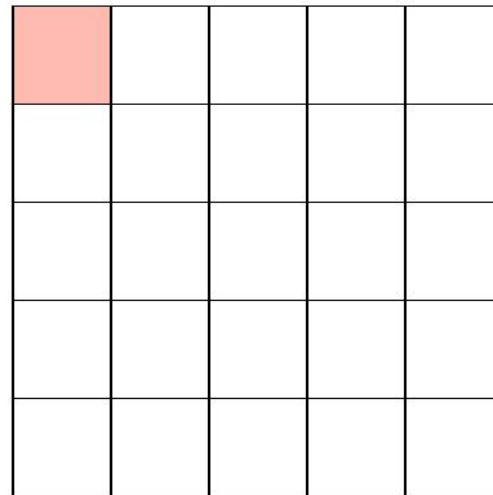
Feature Map

# Stride (Subsample)

[towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2](https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2)

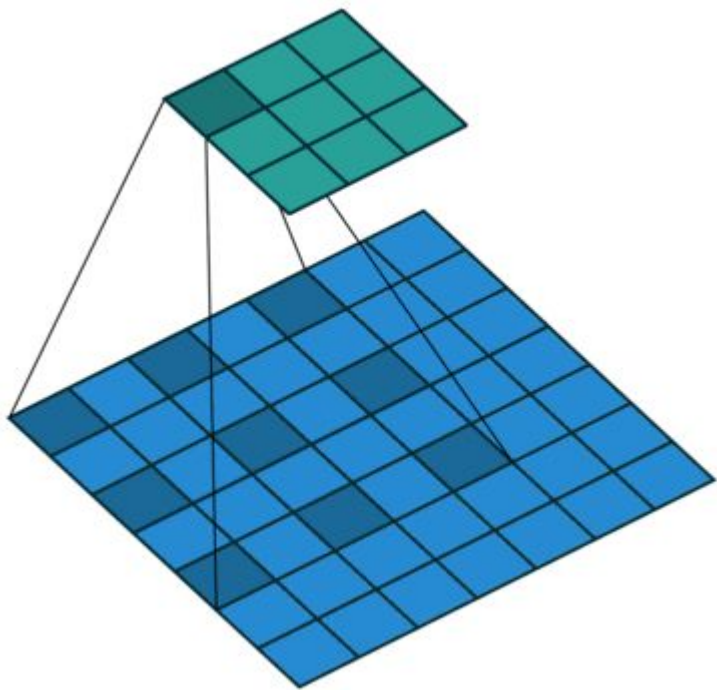


Stride 1 with Padding



Feature Map

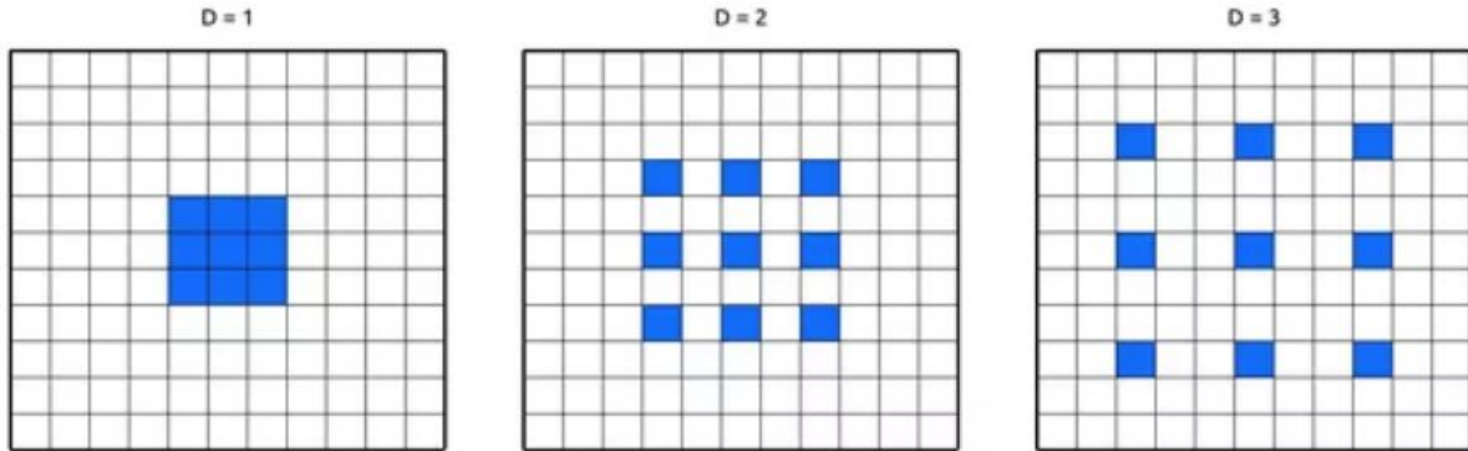
# Dilated Convolution



[github.com/hassony2/inria-research-wiki/wiki/dilated-convolutions-vs-transposed-convolutions](https://github.com/hassony2/inria-research-wiki/wiki/dilated-convolutions-vs-transposed-convolutions)

[github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)

# Dilated Convolution



This is also equivalent to adding 0 in the filters for all places where the input value should be ignored (but seeing it as skipping inputs is closer to the computationally effective approach of the problem)

# Dilated Convolution

In many such applications one wants to integrate information from different spatial scales and balance two properties:

1. local, pixel-level accuracy, such as precise detection of edges, and
2. integrating knowledge of the wider, global context

To address this problem, people often use some kind of multi-scale convolutional neural networks, which often relies on spatial pooling. Instead the authors here propose using layers *dilated convolutions*, which allow us to address the multi-scale problem efficiently without increasing the number of parameters too much.

<https://www.inference.vc/dilated-convolutions-and-kronecker-factorisation/>

# Filters

- A filter is represented by a vector of weights with which we convolve the input.
- A measure for how close a patch of input resembles a feature/pattern.
- Every network layer has filters to if detect specific patterns are present

# Convolution

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

*Typical sliding  
window!!!*



# FEATURE MAPS

- The feature map is the output of one filter applied to the previous layer.

# FEATURE MAPS



Arjen Hiemstra, <https://www.facebook.com/EconomicSingularity/>

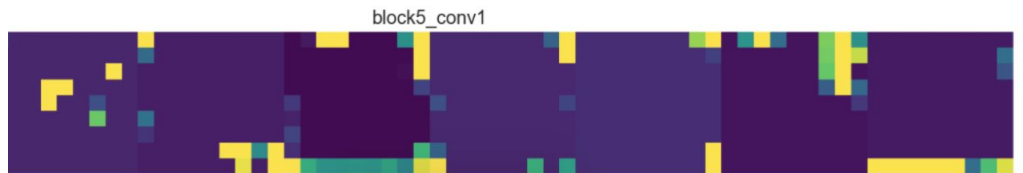
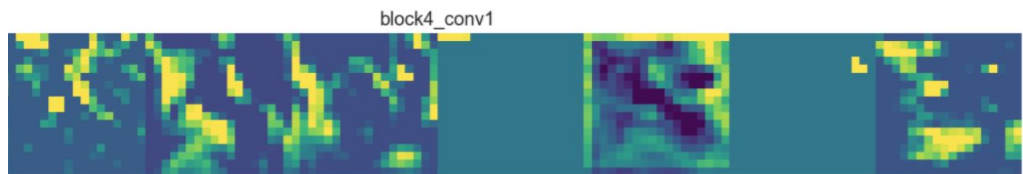
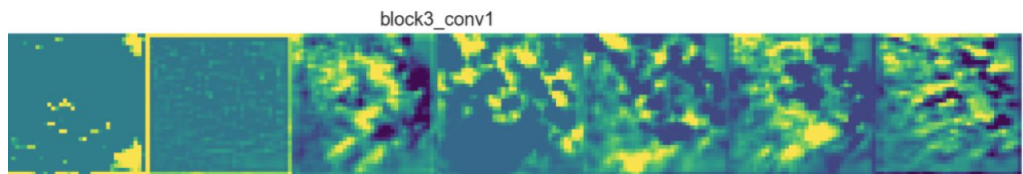
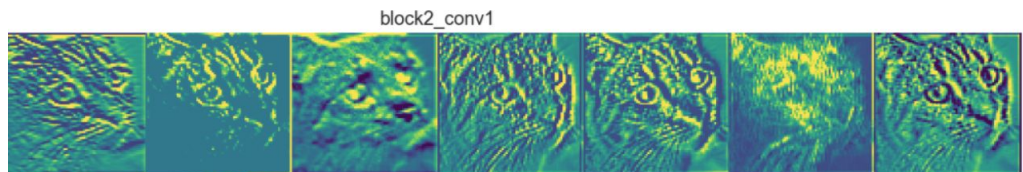
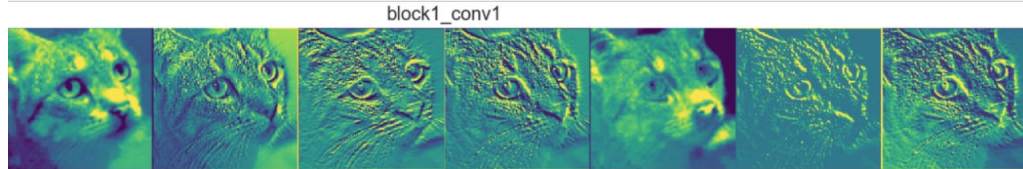
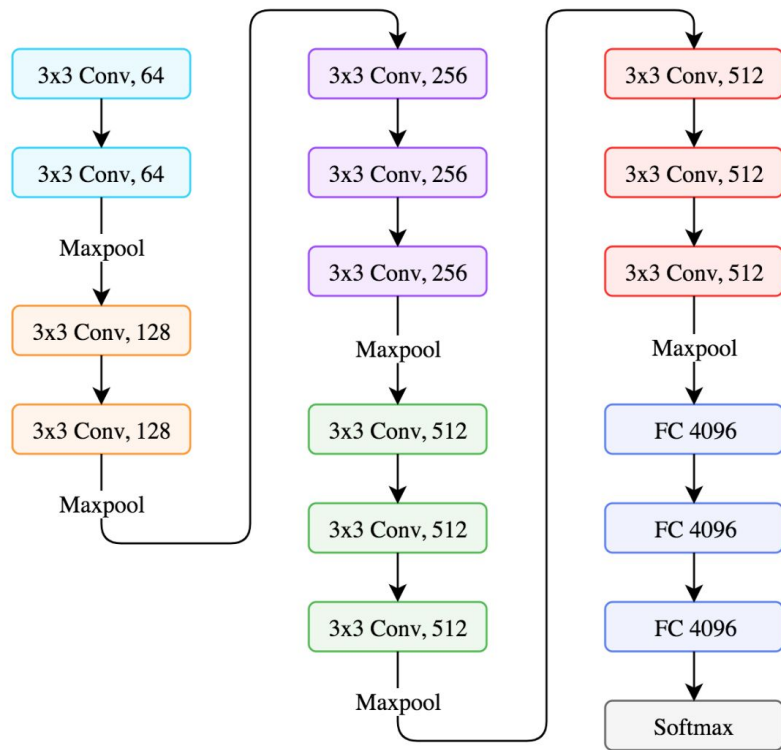
Answered Feb 27, 2017

For example, your first feature map could for example look for curves. The next feature map could look at a combination of curves that build circles. The next feature map could detect a bicycle from lines and circle features.

I found this research document lists this very clearly:

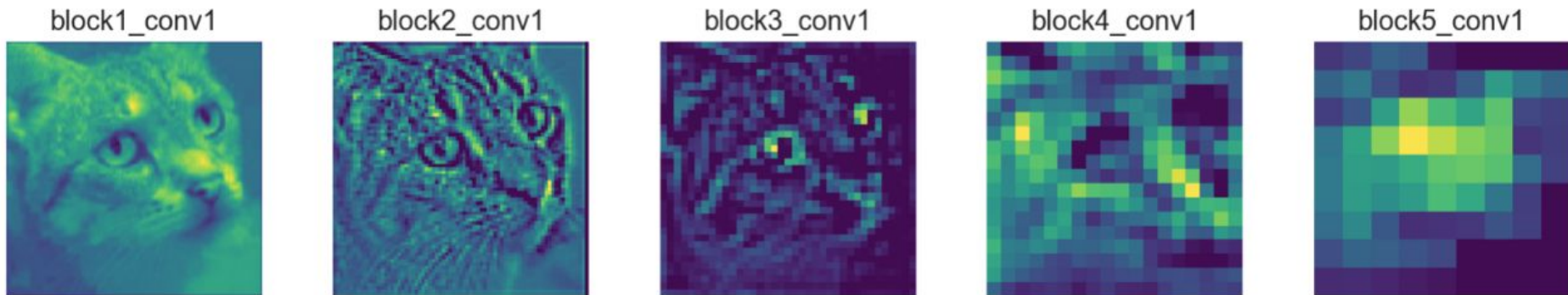
[\[1311.2901\] Visualizing and Understanding Convolutional Networks](#) ↗

# Layer Visualization



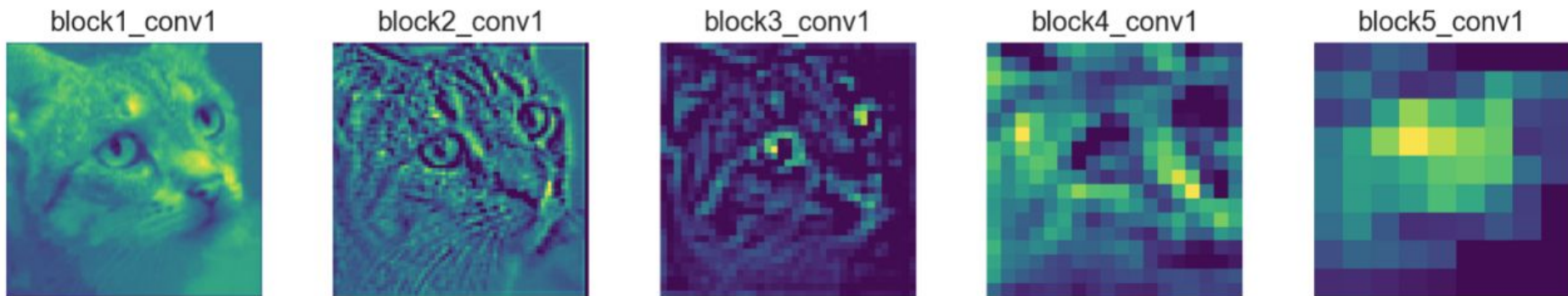
## Layer Visualization (First few layers)

- Retain most of the information (acts edge detectors, simple shapes detectors)



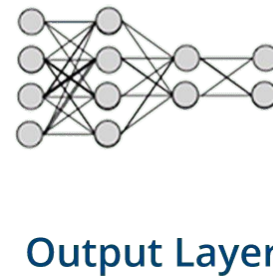
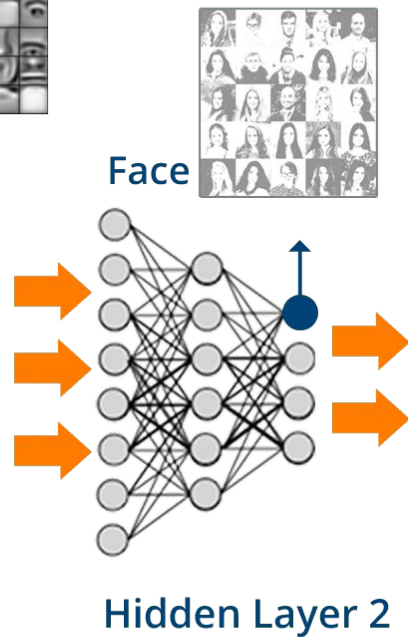
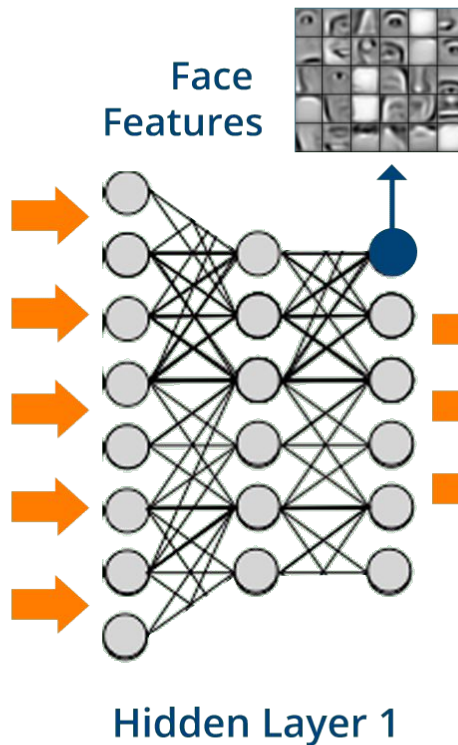
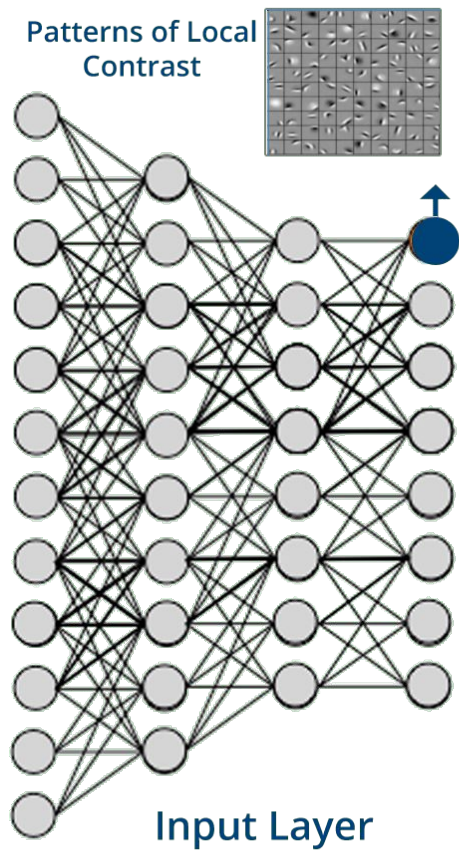
## Layer Visualization (Deeper layers)

- feature maps less like the original image more like an abstract representation, less recognizable
- combine features detected in the more shallow layers
- high level concepts like “cat nose” or “dog ear”, why feature maps go blank



[towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-nets-works-584bc134c1e2](https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-nets-works-584bc134c1e2)





# HIDDEN LAYER VISUALIZATION

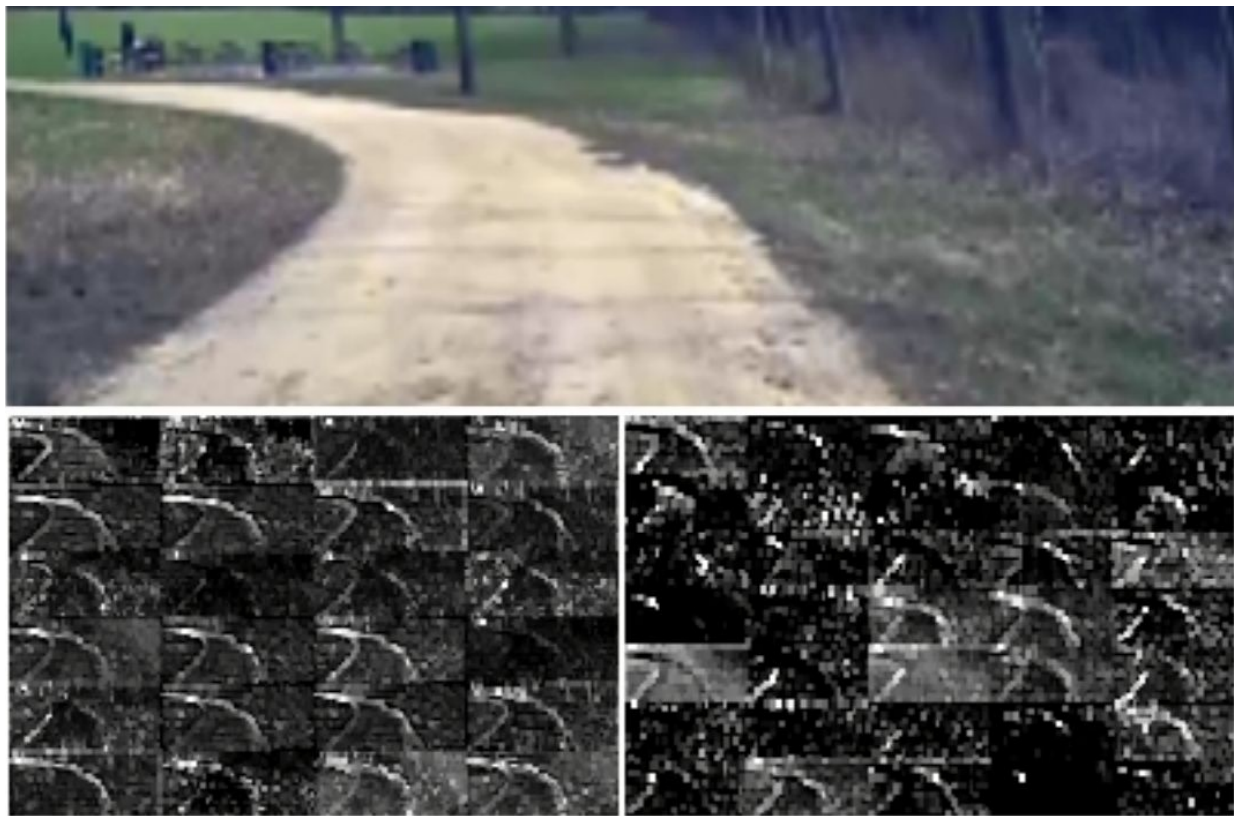
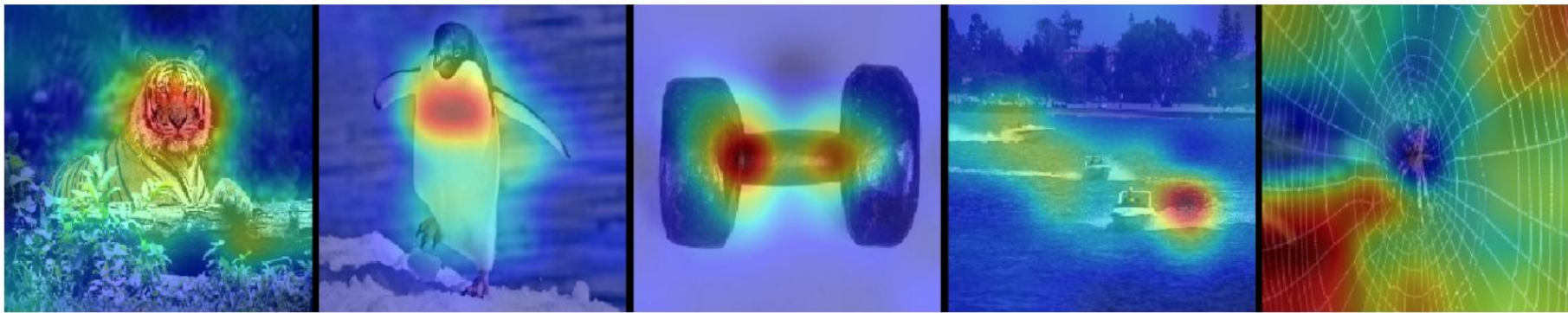


Figure 7: How the CNN “sees” an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

# Layer Visualization (Attention)

## Attention Maps



*How can we assess whether a network is attending to correct parts of the image in order to generate a decision?*

Use this library! [github.com/raghakot/keras-vis](https://github.com/raghakot/keras-vis)



# Layer Visualization (Attention)

raghakot / keras-vis

Watch

64

Code

Issues 59

Pull requests 4

Projects 0

Wiki

Insights

Branch: master

keras-vis / applications / self\_driving /

Create new

raghakot - Introducing grad\_modifiers, backprop\_modifiers, input\_modifiers. ...

..

images update self driving example

README.md - Introducing grad\_modifiers, backprop\_modifiers, input\_modifiers.

model.py Major changes.

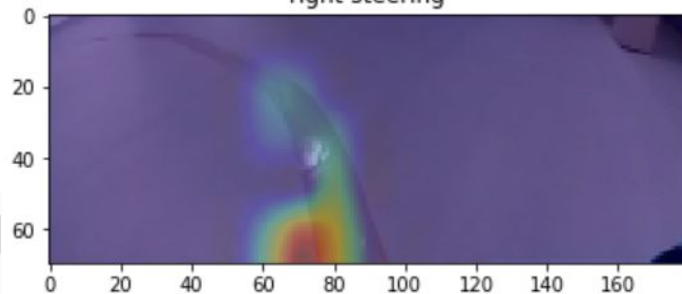
visualize\_attention.ipynb - Introducing grad\_modifiers, backprop\_modifiers, input\_modifiers.

weights.hdf5 Major changes.

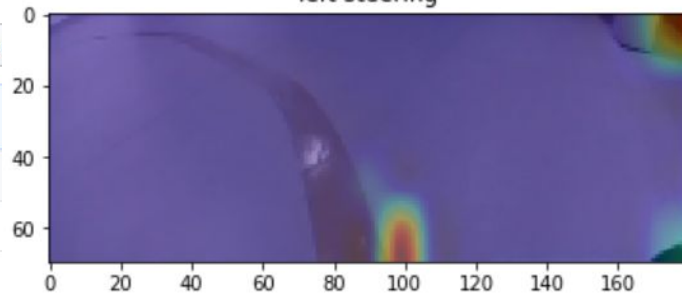
README.md

Use this library! [github.com/raghakot/keras-vis](https://github.com/raghakot/keras-vis)

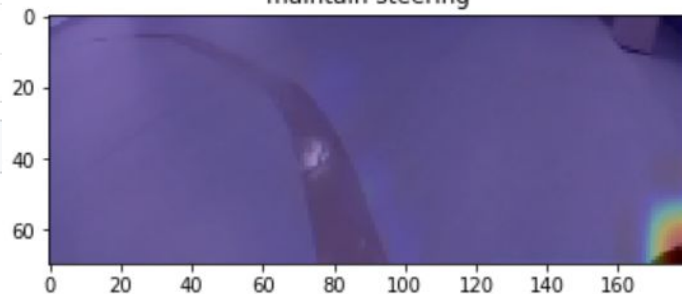
right steering



left steering



maintain steering



# Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization

<https://arxiv.org/pdf/1610.02391v1.pdf>

'boxer' (243 or 242 in keras)



'tiger cat' (283 or 282 in keras)



# LAYER VISUALIZATION

**Jacob Gil's work visualizing attention using Grad Cam in Keras**

<https://github.com/jacobgil/keras-grad-cam>

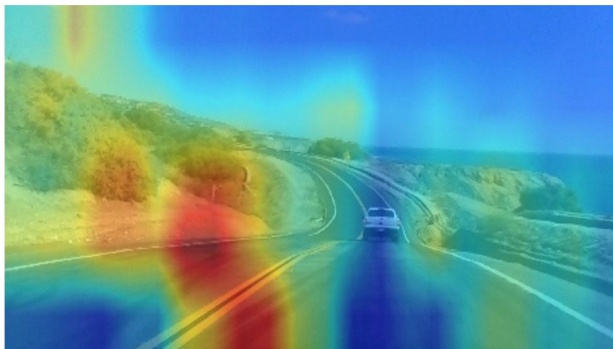
**Class activation maps in Keras for visualizing where deep learning networks pay attention**

<https://jacobgil.github.io/deep-learning/class-activation-maps>

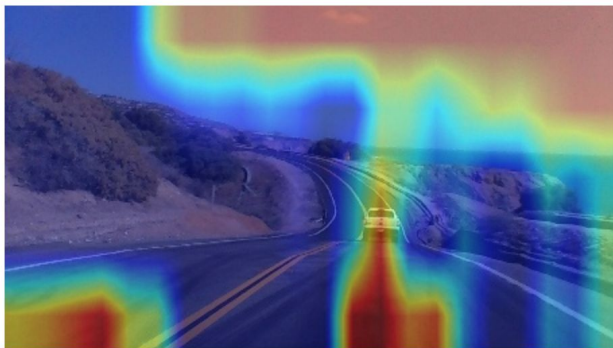
**Visualizations for regressing wheel steering angles in self driving cars**

<https://jacobgil.github.io/deep-learning/vehicle-steering-angle-visualizations>

Lets look at an example. For the same image, we could target pixels that contribute to steering right:



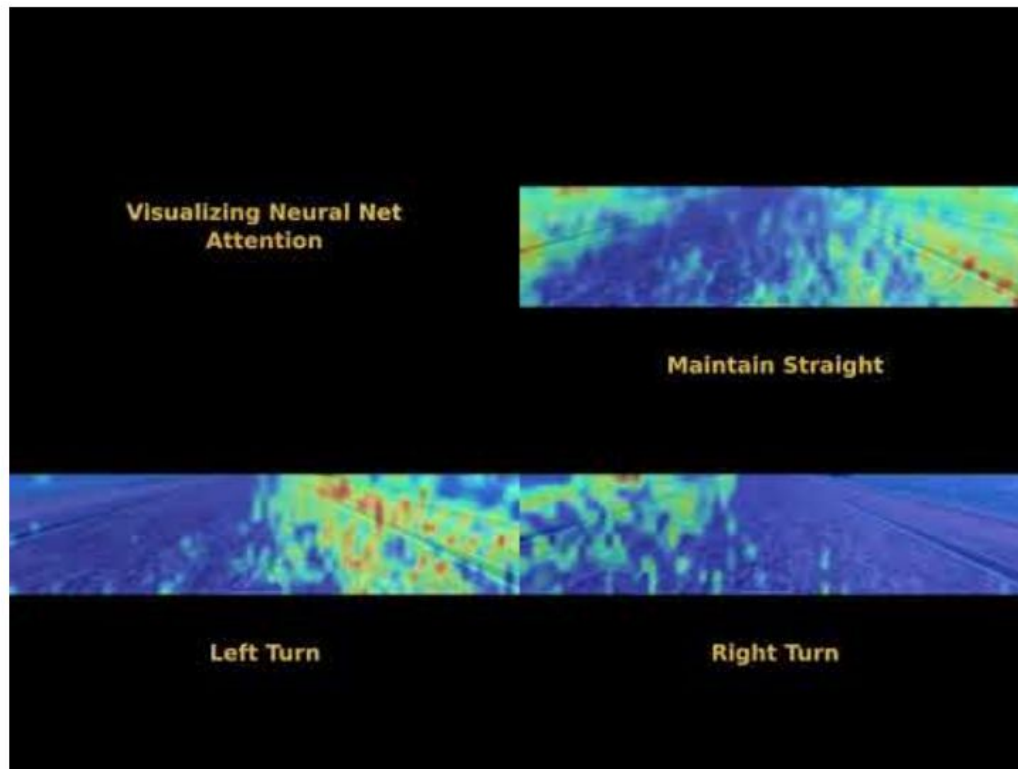
And we could also target pixels that contribute to steering to the center:



# LAYER VISUALIZATION

SEE CODE

[https://github.com/andrewsommerlot/Carnd-Behavioral-Cloning/blob/master/visualize\\_attention.py](https://github.com/andrewsommerlot/Carnd-Behavioral-Cloning/blob/master/visualize_attention.py)



# Layer Visualization (Other Links, with code)

## **Understanding Neural Networks Through Deep Visualization**

<http://yosinski.com/deepvis>

## **Visualizing parts of Convolutional Neural Networks using Keras and Cats**

<https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59>

Erik Kreppel - Jan 23, 2017

## **Visualizing Convolutional Neural Networks in Python**

Faizan Shaikh - March 22, 2018

<https://www.analyticsvidhya.com/blog/2018/03/essentials-of-deep-learning-visualizing-convolutional-neural-networks/>

## **Other list of links**

<https://github.com/mrgloom/CNN-heatmap>

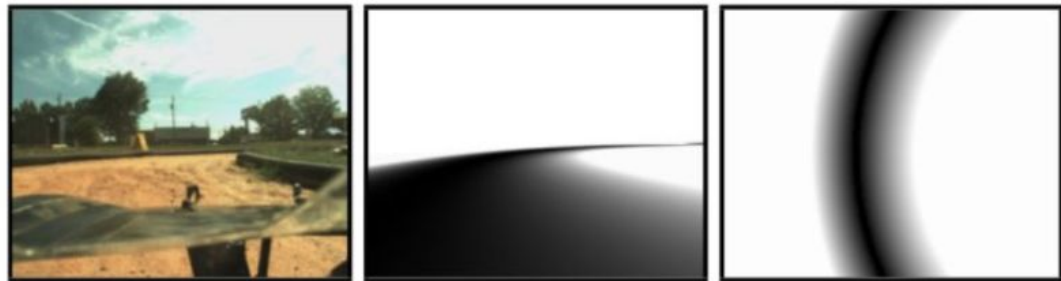
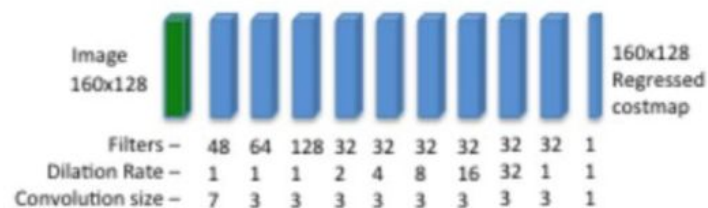
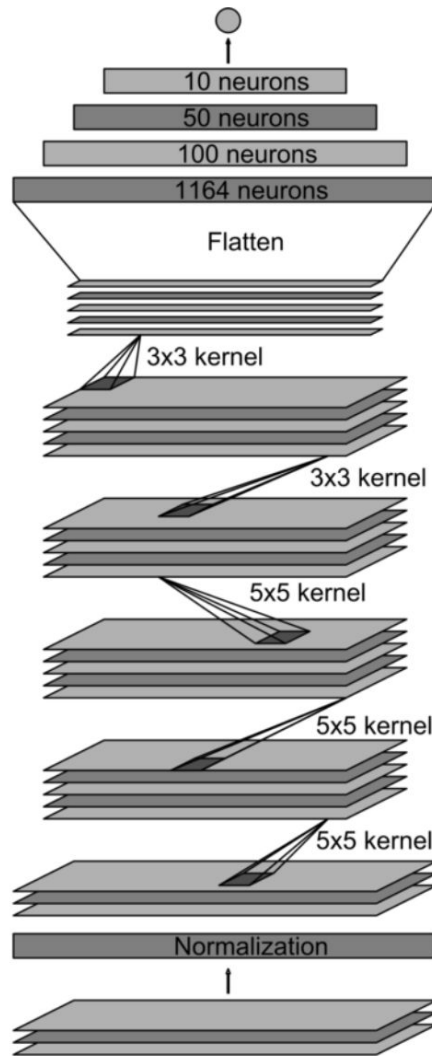


Figure 2: Network architecture with input and training targets. Left: Neural network architecture used to produce top down cost maps. Right: example input image, image plane training target and top down training target, respectively

# NETWORK ARCHITECTURE

- 9 layers,
- 1 normalization layer
- 5 convolutional layers
- 3 fully connected layers.

From:  
<https://devblogs.nvidia.com/deep-learning-self-driving-cars/>



Output: vehicle con...

Fully-connected lay

Fully-connected lay

Fully-connected lay

Convolutional

feature map

64@1x18

Convolutional

feature map

64@3x20

Convolutional

feature map

48@5x22

Convolutional

feature map

36@14x47

Convolutional

feature map

24@31x98

Normalized

input planes

3@66x200

Input planes

3@66x200



```
def build_modified_nvidia_model():
    model = Sequential()
    model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))
    model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))
    model.add(Dense(1))
    model.compile(optimizer = OPTIMIZER_TYPE, loss = LOSS_TYPE)
    return model
```



# A Review of Machine Learning Intuitions

- Activation Functions
- Epoch
- Underfit vs Overfit
- Overfitting Analogy and Dropouts
- Learning Rate
- Optimizers
- Batch Size
- Steps per Epoch
- Convolution, Dilation, Stride (subsample) , Filters (kernel)
- Layer visualization