

Lecture 8: Visualizing and Understanding

Administrative

- Assignment 2 due tonight
- Assignment 1 + Quiz 1 grades released
- Assignment 3 released (Due Nov 5)
- Quiz 2 this friday (Practice Quiz Posted)
- Google Cloud Credits posted to EdStem

Choosing Hyperparameters

Choosing Hyperparameters: Grid Search

Choose several values for each hyperparameter
(Often space choices log-linearly)

Example:

Weight decay: $[1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}]$

Learning rate: $[1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}]$

Evaluate all possible choices on this hyperparameter grid

Choosing Hyperparameters: Random search

Choose several values for each hyperparameter
(Often space choices log-linearly)

Example:

Weight decay: **log-uniform** on $[1 \times 10^{-4}, 1 \times 10^{-1}]$

Learning rate: **log-uniform** on $[1 \times 10^{-4}, 1 \times 10^{-1}]$

Run many different trials

Random Search vs. Grid Search

*Random Search for
Hyper-Parameter Optimization*
Bergstra and Bengio, 2012

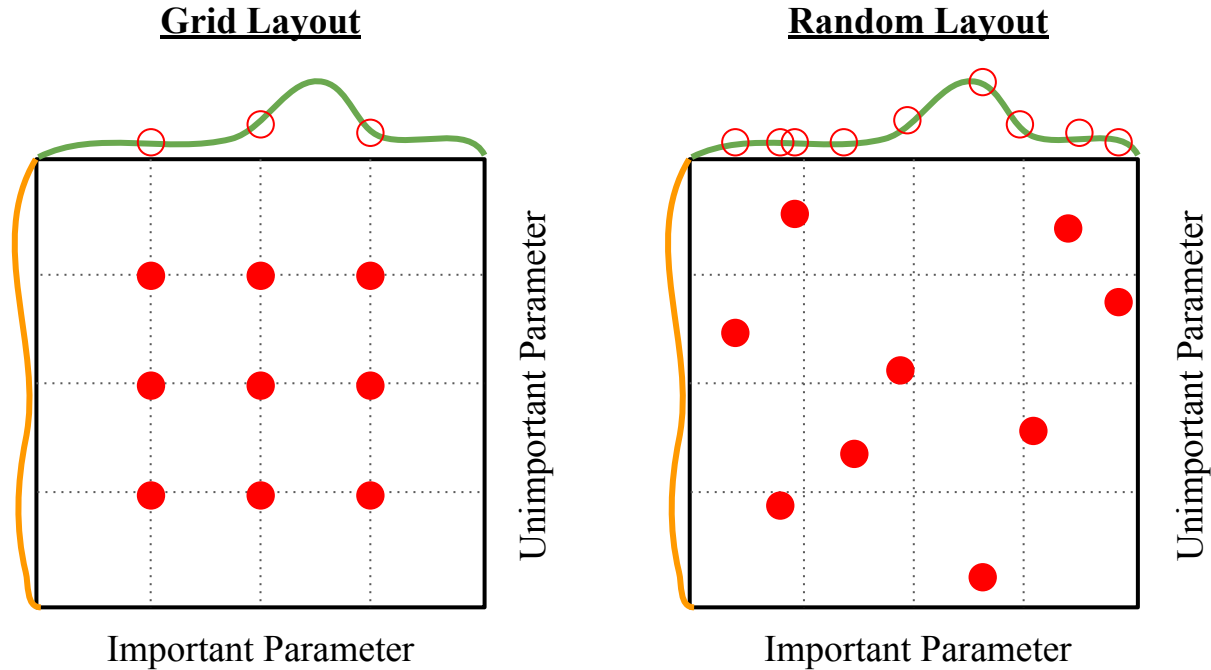


Illustration of Bergstra et al., 2012 by Shayne Longpre, copyright CS231n 2017

Choosing Hyperparameters

(without tons of GPUs)

Choosing Hyperparameters

Step 1: Check initial loss

Turn off weight decay, sanity check loss at initialization
e.g. $\log(C)$ for softmax with C classes

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Try to train to 100% training accuracy on a small sample of training data (~5-10 minibatches); fiddle with architecture, learning rate, weight initialization

Loss not going down? LR too low, bad initialization

Loss explodes to Inf or NaN? LR too high, bad initialization

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Use the architecture from the previous step, use all training data, turn on small weight decay, find a learning rate that makes the loss drop significantly within ~ 100 iterations

Good learning rates to try: $1e-1$, $1e-2$, $1e-3$, $1e-4$

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Step 4: Coarse grid, train for ~1-5 epochs

Choose a few values of learning rate and weight decay around what worked from Step 3, train a few models for ~1-5 epochs.

Good weight decay to try: $1e-4$, $1e-5$, 0

Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Step 4: Coarse grid, train for ~1-5 epochs

Step 5: Refine grid, train longer

Pick best models from Step 4, train them for longer (~10-20 epochs) without learning rate decay

Choosing Hyperparameters

Step 1: Check initial loss

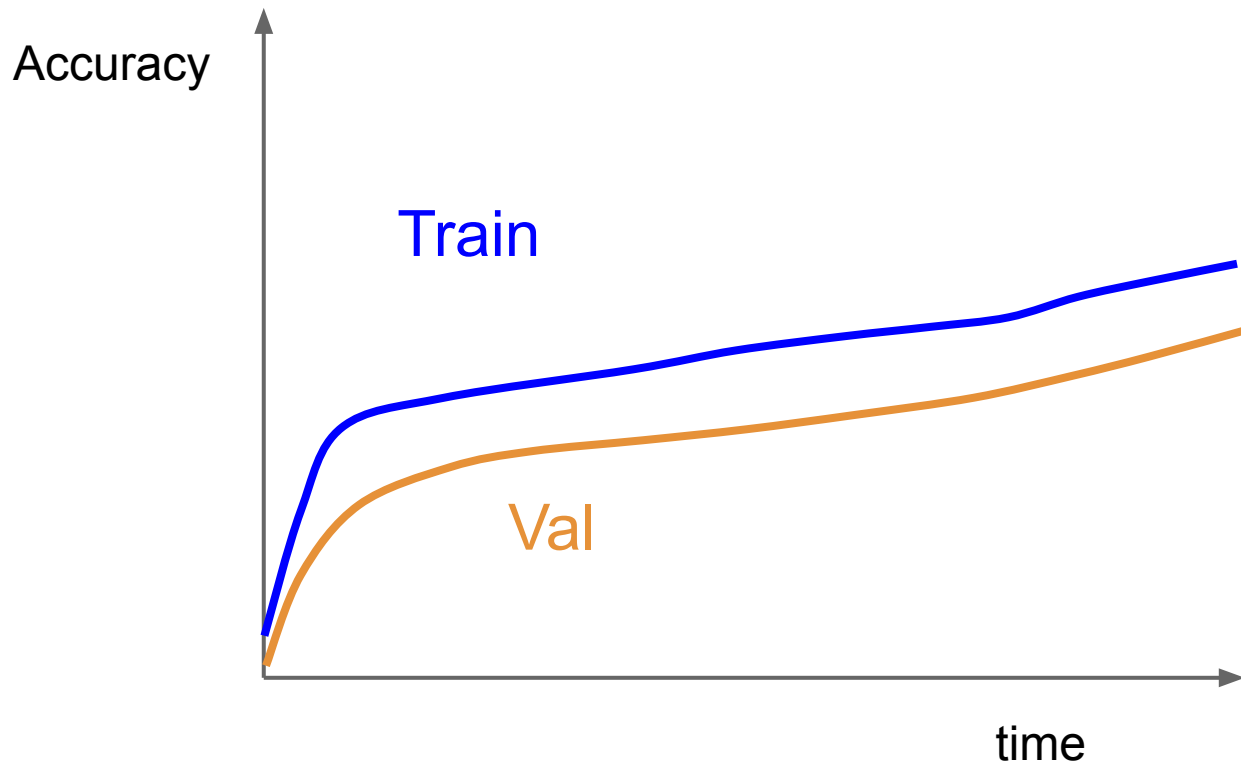
Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

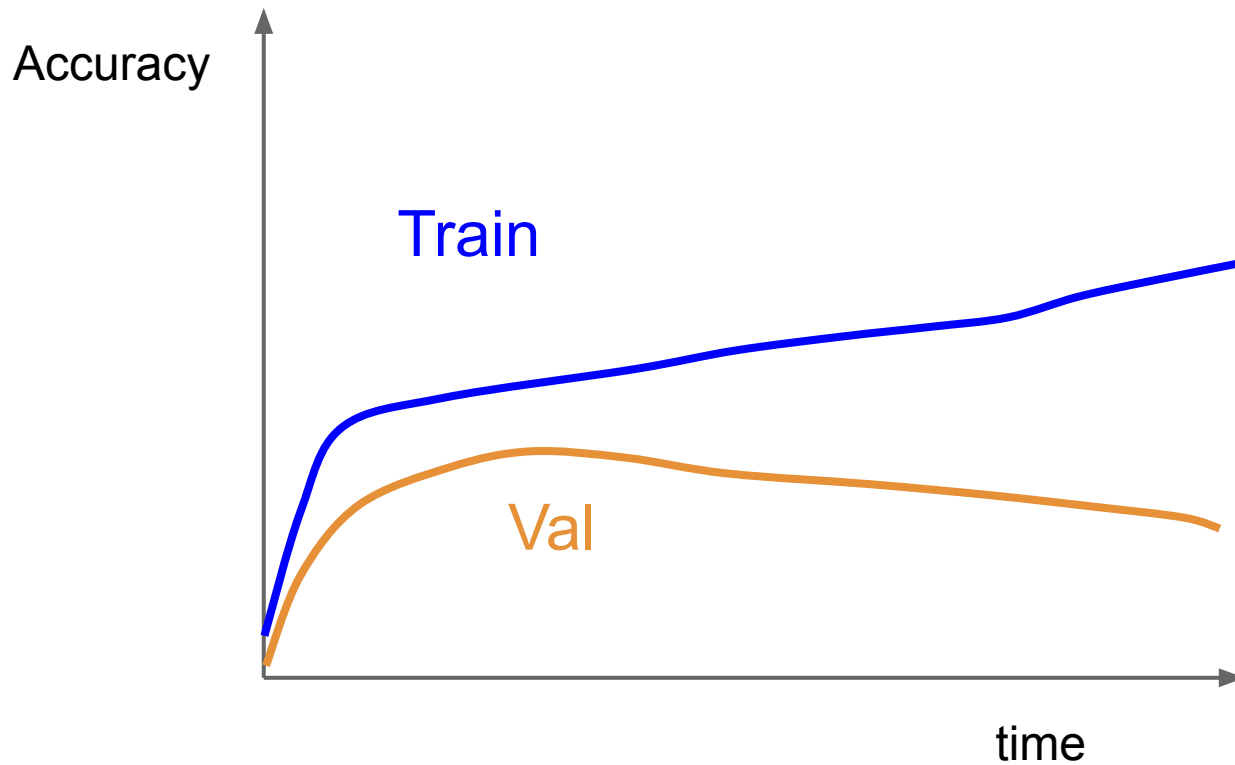
Step 4: Coarse grid, train for ~1-5 epochs

Step 5: Refine grid, train longer

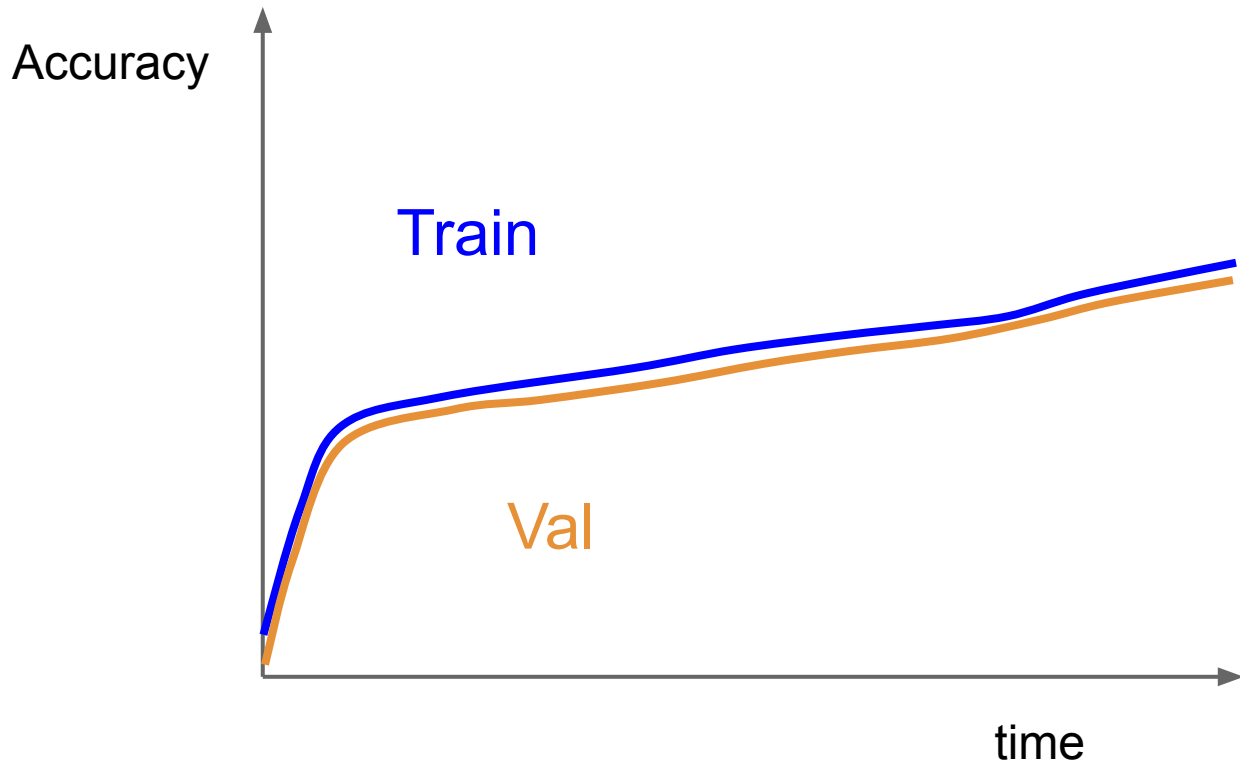
Step 6: Look at loss and accuracy curves



Q1. You see this. What should you do?



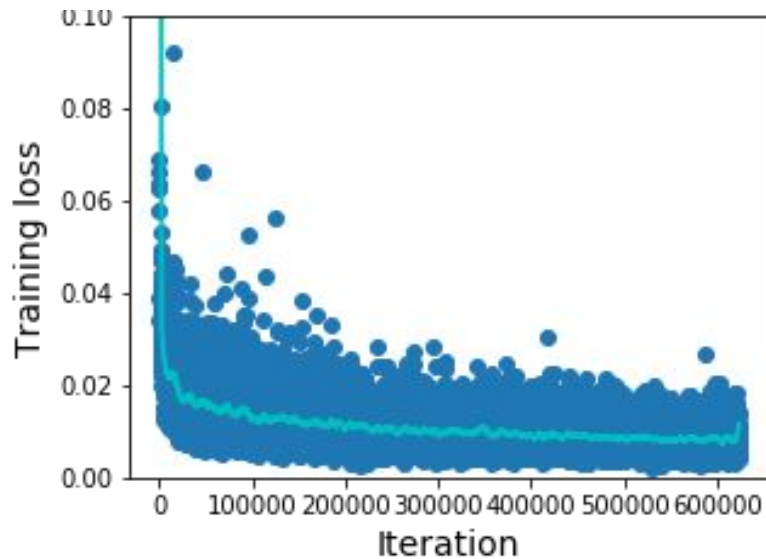
Q2. You see this. What should you do?



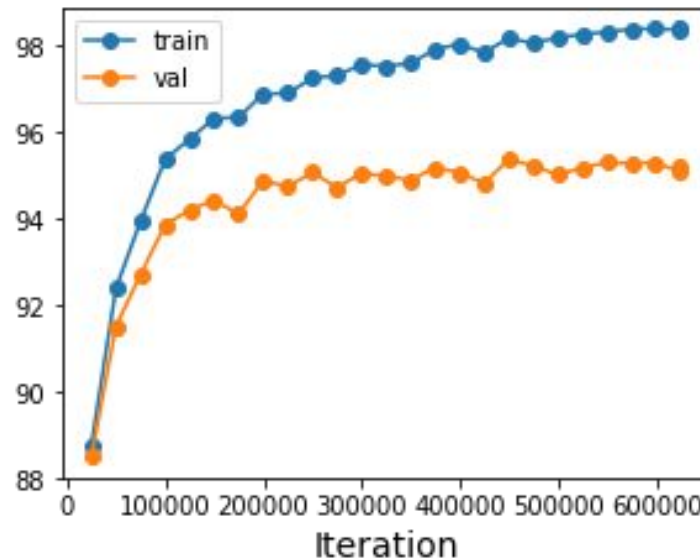
Q3. You see this. What should you do?

Look at learning curves!

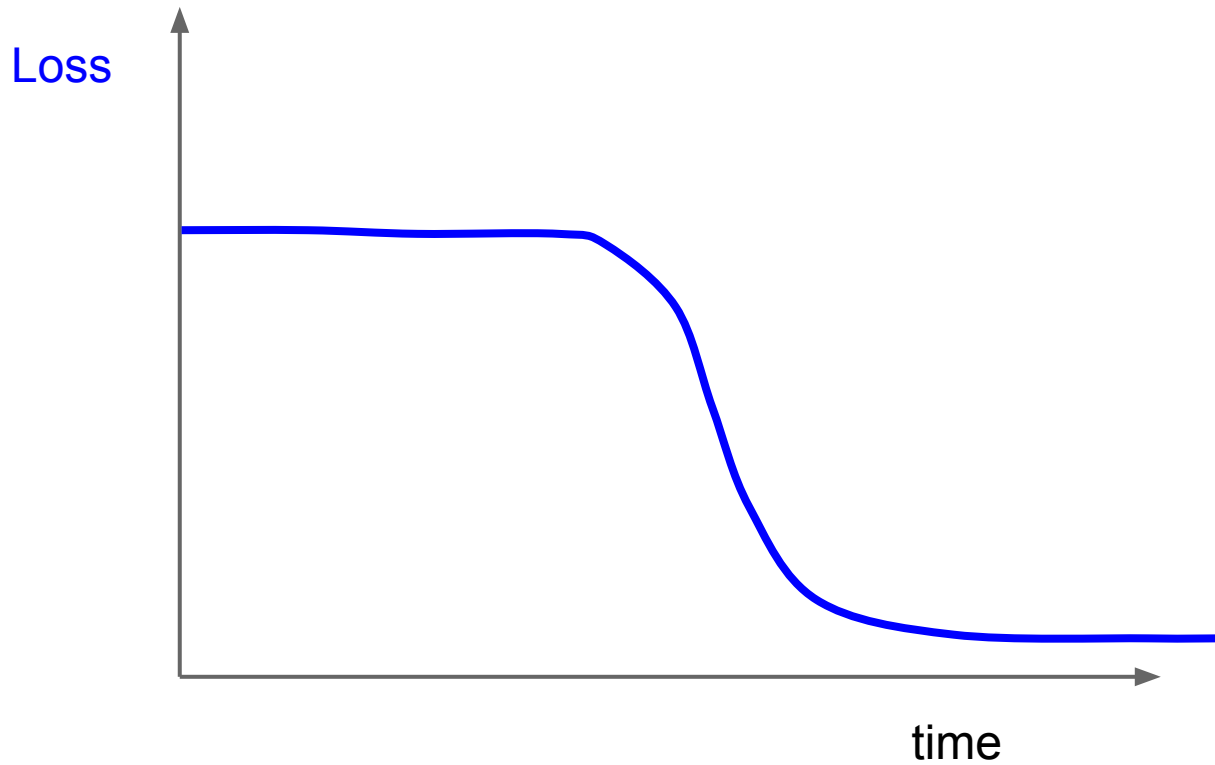
Training Loss



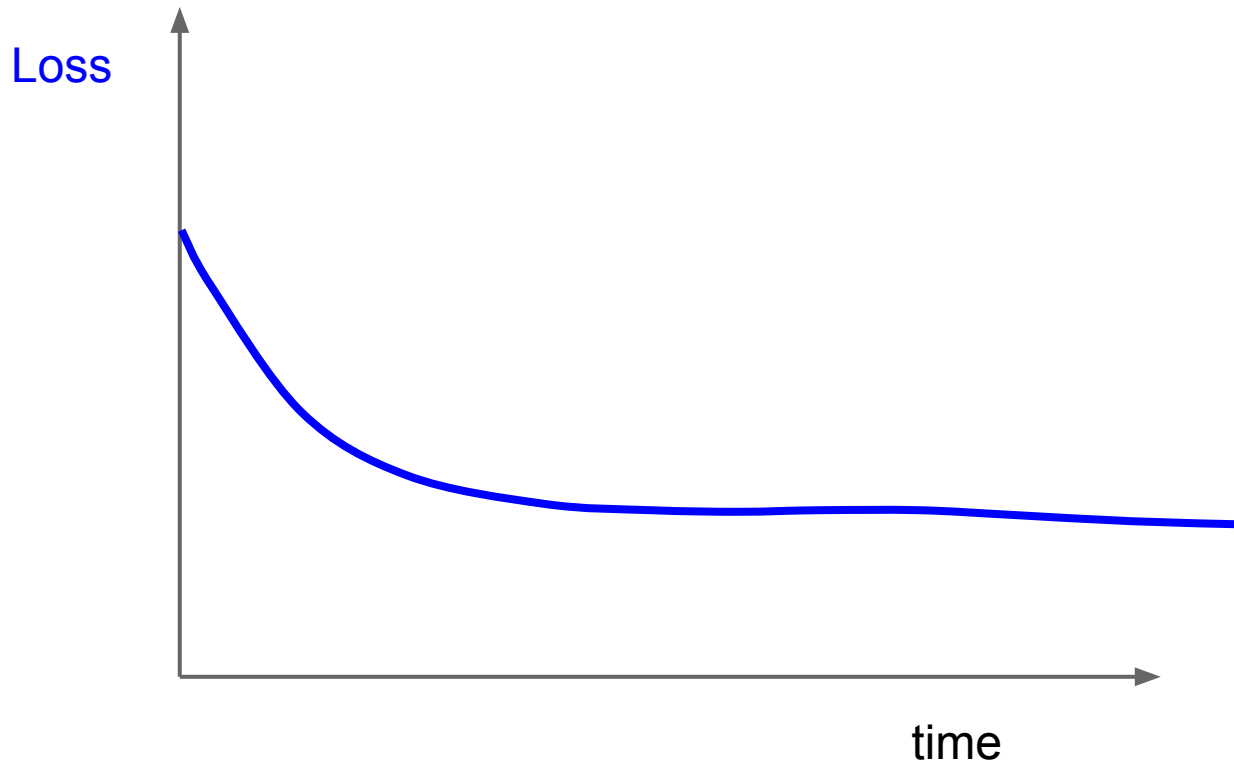
Train / Val Accuracy



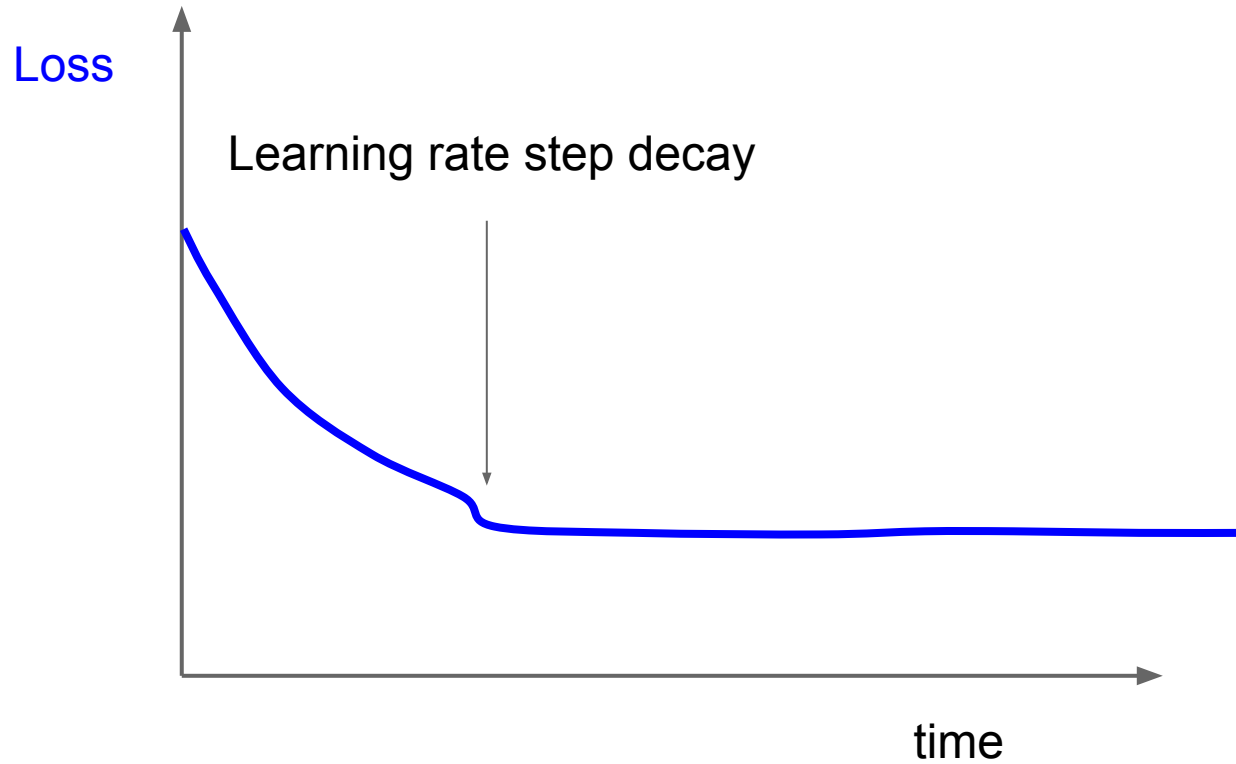
Losses may be noisy, use a scatter plot and also plot moving average to see trends better



Q4. You see this. What should you do?



Q5. You see this. What should you do?

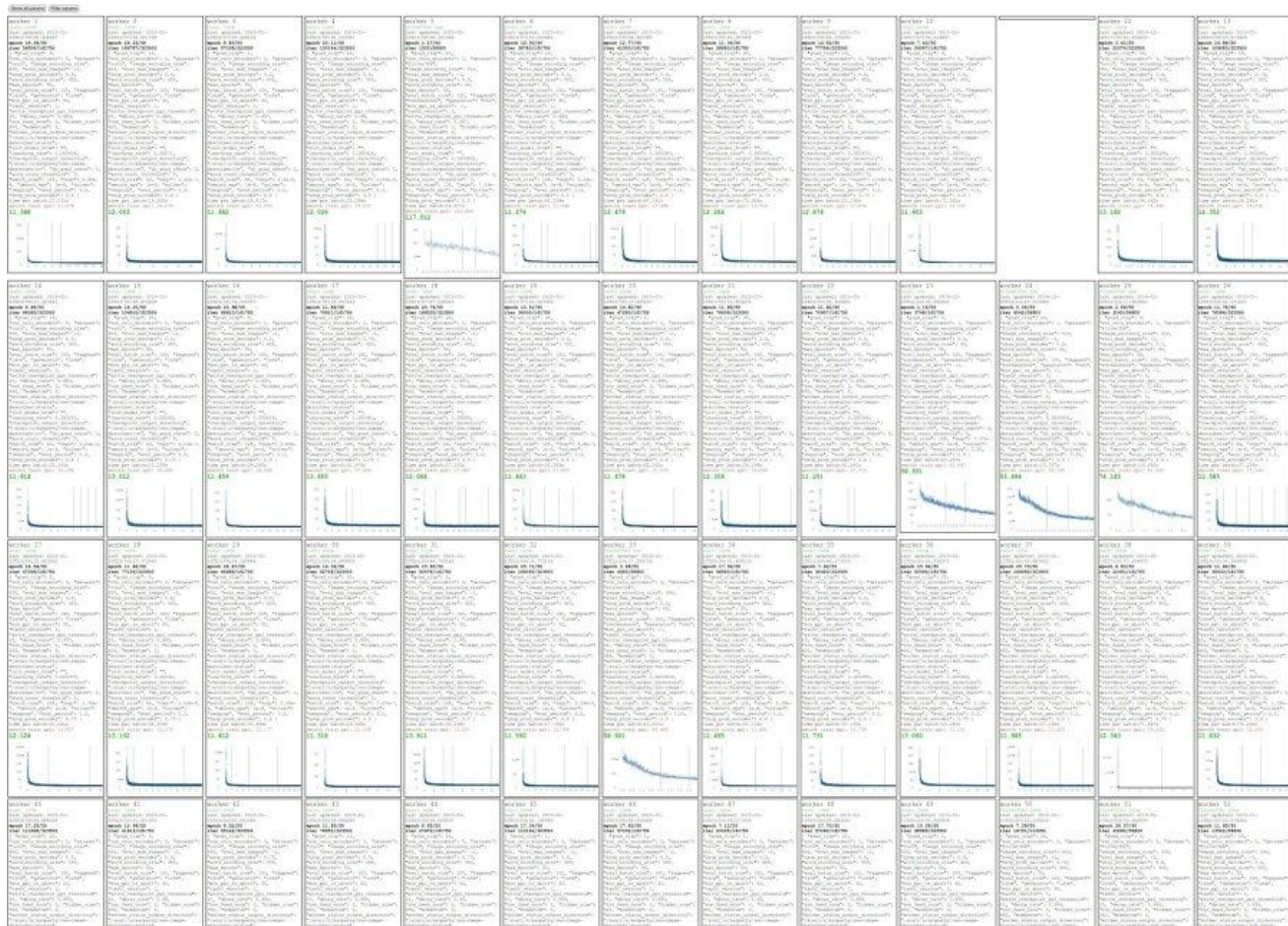


Q5. You see this. What should you do?

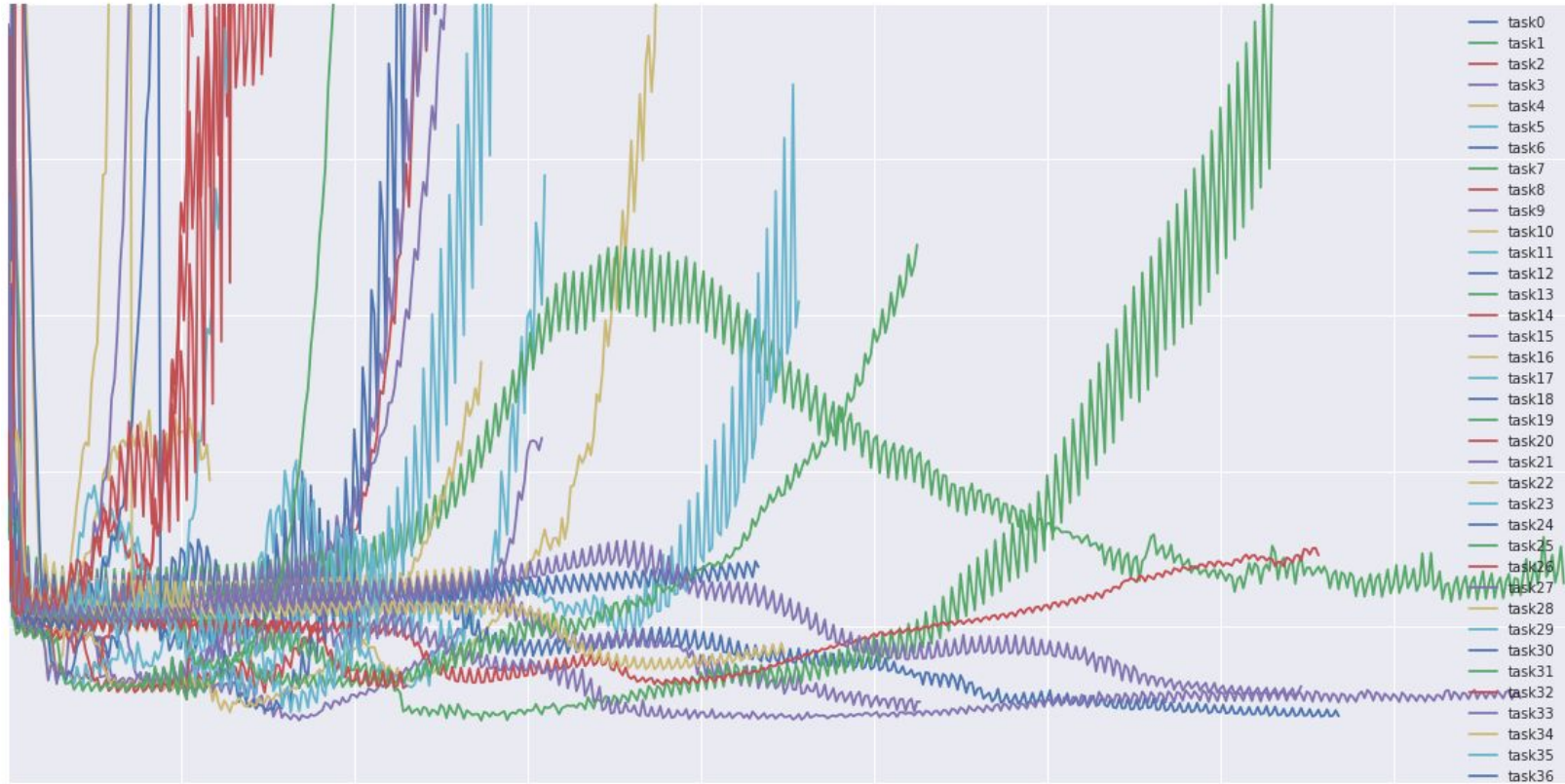
Cross-validation

We develop "command centers" to visualize all our models training with different hyperparameters

check out [weights](#) and [biases](#)



You can plot all your loss curves for different hyperparameters on a single plot



Choosing Hyperparameters

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down

Step 4: Coarse grid, train for ~1-5 epochs

Step 5: Refine grid, train longer

Step 6: Look at loss and accuracy curves

Step 7: GOTO step 5

Hyperparameters to play with:

- learning rate,
- Its decay schedule, update type
- regularization (L2/Dropout strength)

[This image](#) by Paolo Guereta is licensed under [CC-BY 2.0](#)

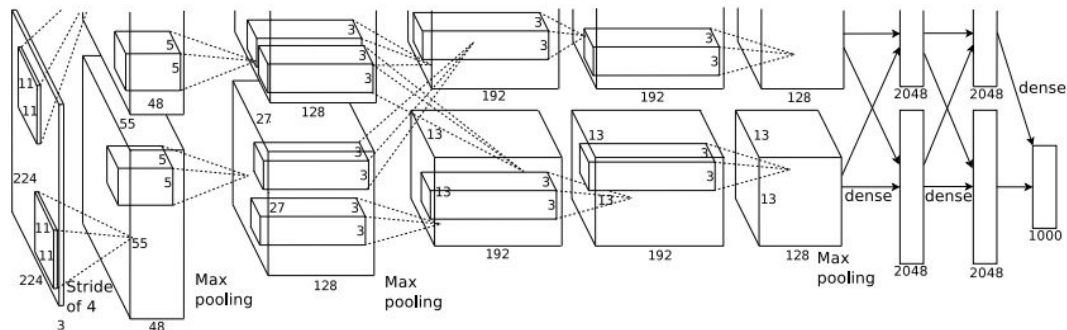
Summary

- Improve your training error:
 - Optimizers
 - Learning rate schedules
- Improve your test error:
 - Regularization
 - Choosing Hyperparameters

Visualizing and Understanding

Today: What's going on inside ConvNets?

This image is CC0 public domain



Class Scores:
1000 numbers

Input Image:
3 x 224 x 224

↑ ↑ ↑ ↑ ↑ ↑ ↑
What are the intermediate features looking for?

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figure reproduced with permission.

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

Today's agenda

Visualizing what models have learned:

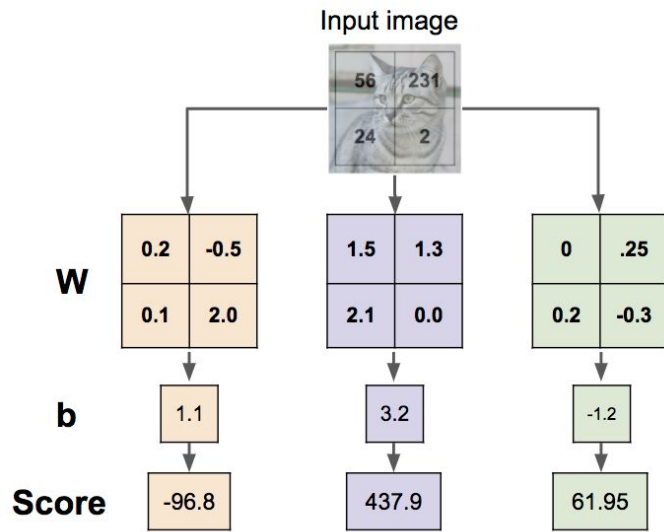
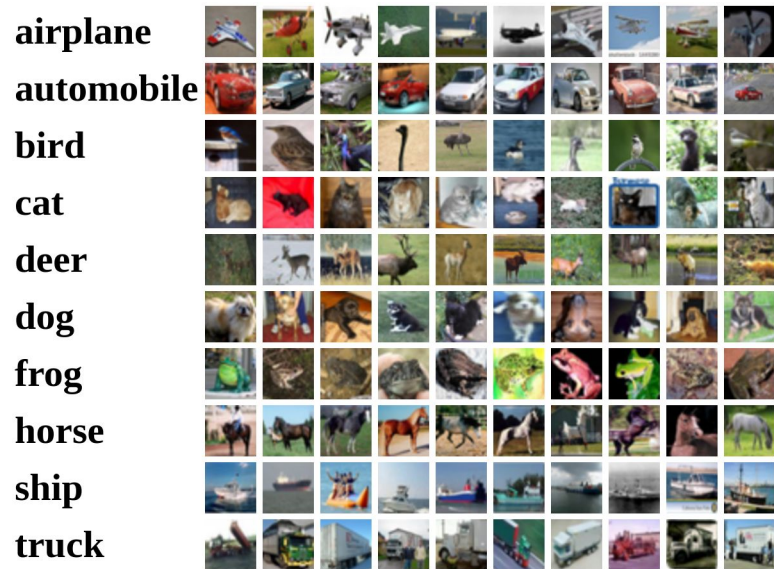
- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

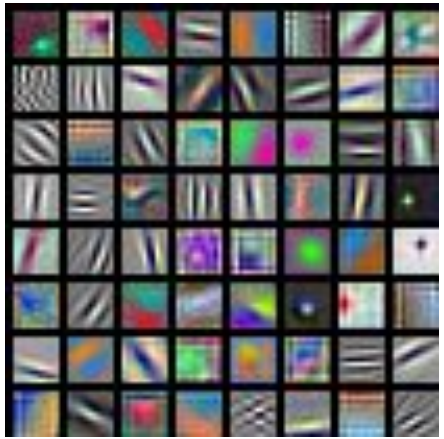
- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

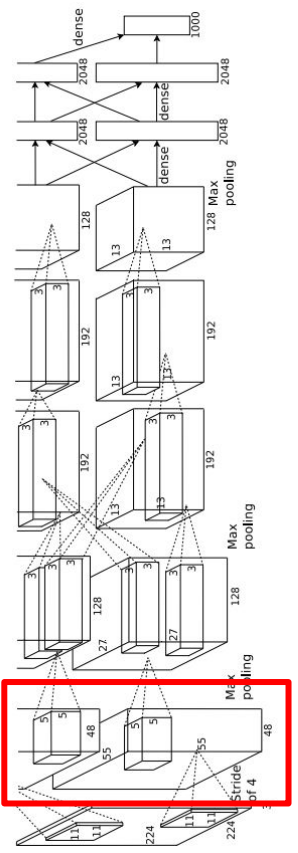
Interpreting a Linear Classifier: Visual Viewpoint



First Layer: Visualize Filters

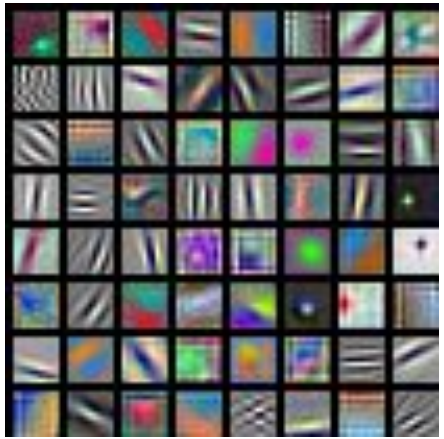


AlexNet:
64 x 3 x 11 x 11



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

First Layer: Visualize Filters



AlexNet:
64 x 3 x 11 x 11



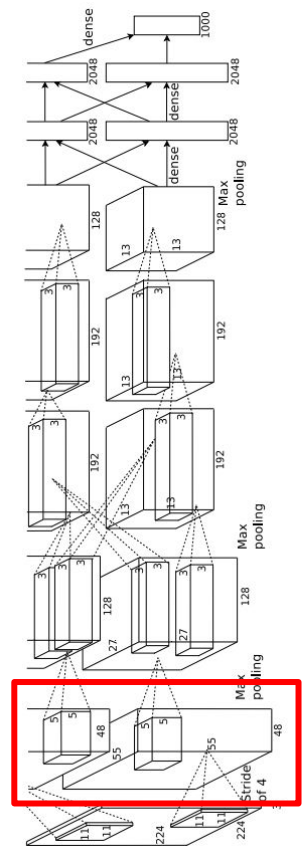
ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
 He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
 Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

Visualize the filters/kernels (raw weights)

We can visualize filters at higher layers, but not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)



layer 1 weights

$16 \times 3 \times 7 \times 7$



layer 2 weights

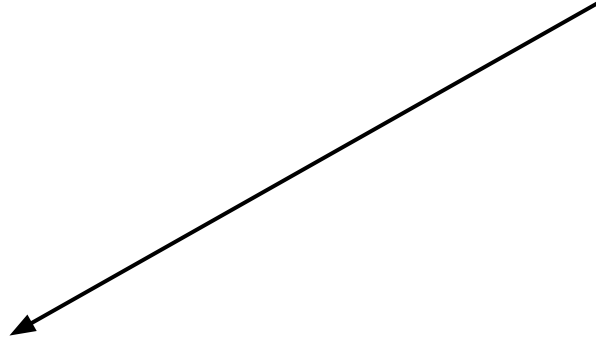
$20 \times 16 \times 7 \times 7$



layer 3 weights

$20 \times 20 \times 7 \times 7$

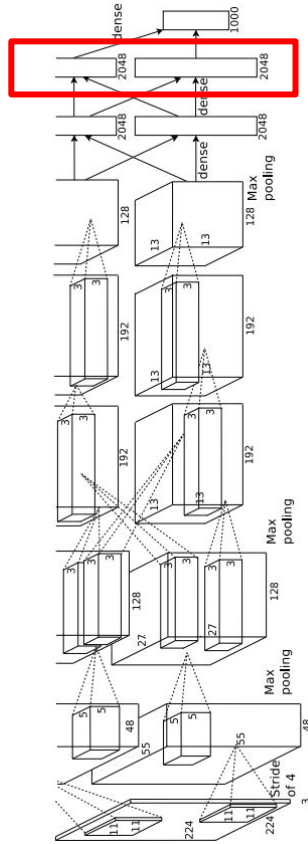
Last Layer



4096-dimensional feature vector for an image
(layer immediately before the classifier)

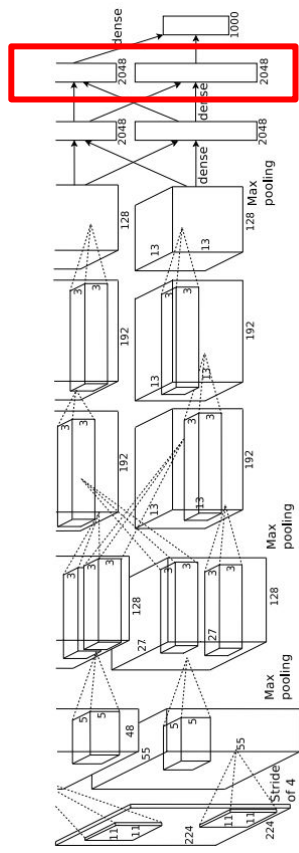
Run the network on many images, collect the
feature vectors

FC7 layer



Last Layer: Nearest Neighbors

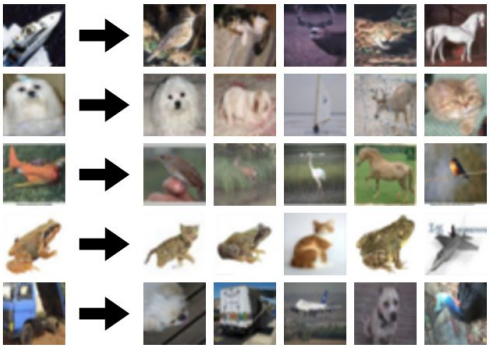
4096-dim vector



Test image L2 Nearest neighbors in feature space

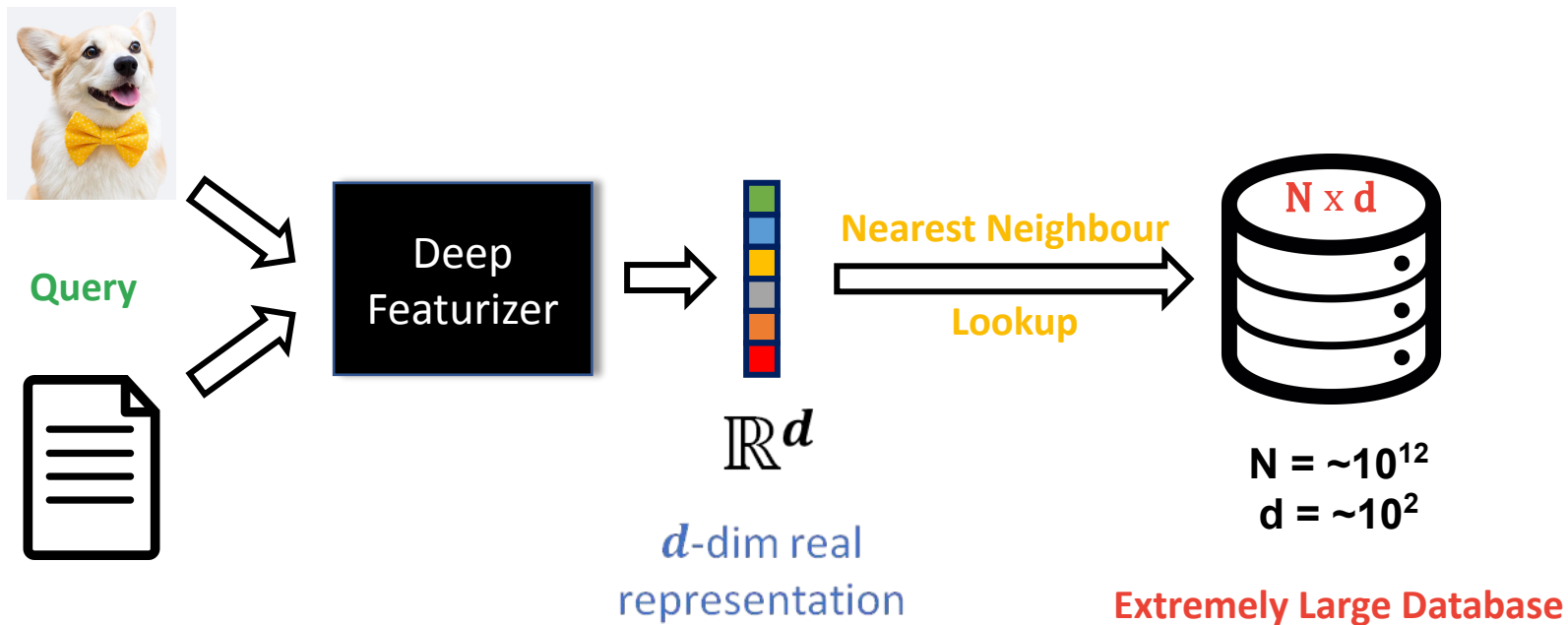


Recall: Nearest neighbors in pixel space

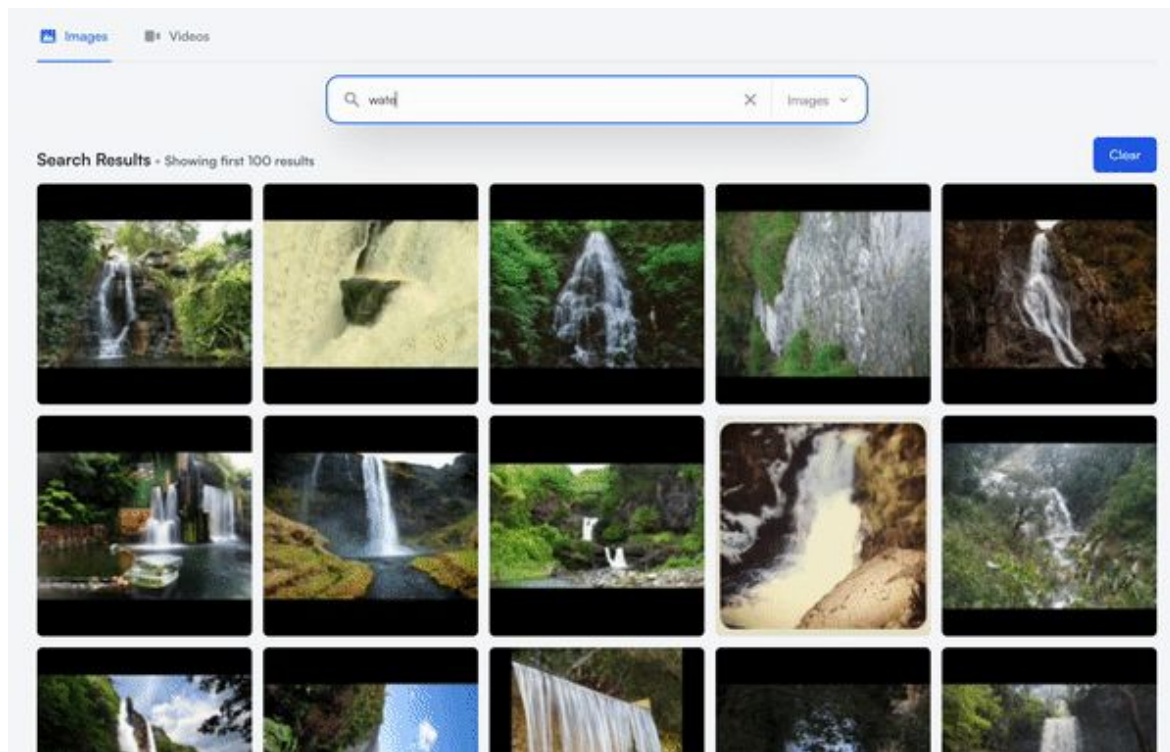


Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012. Figures reproduced with permission.

Last Layer: Learned Metric for “Semantic” Search



Last Layer: Modern Day Search



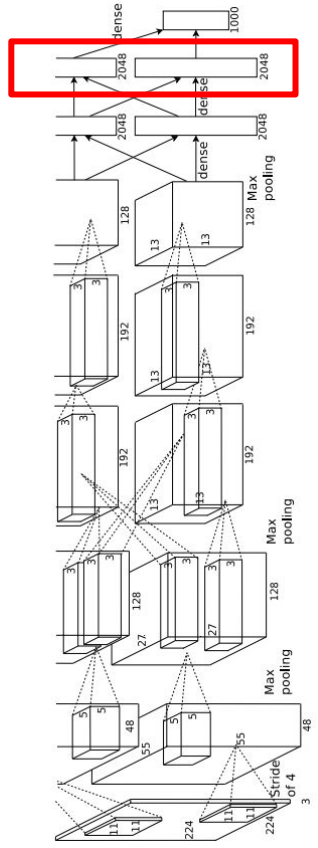
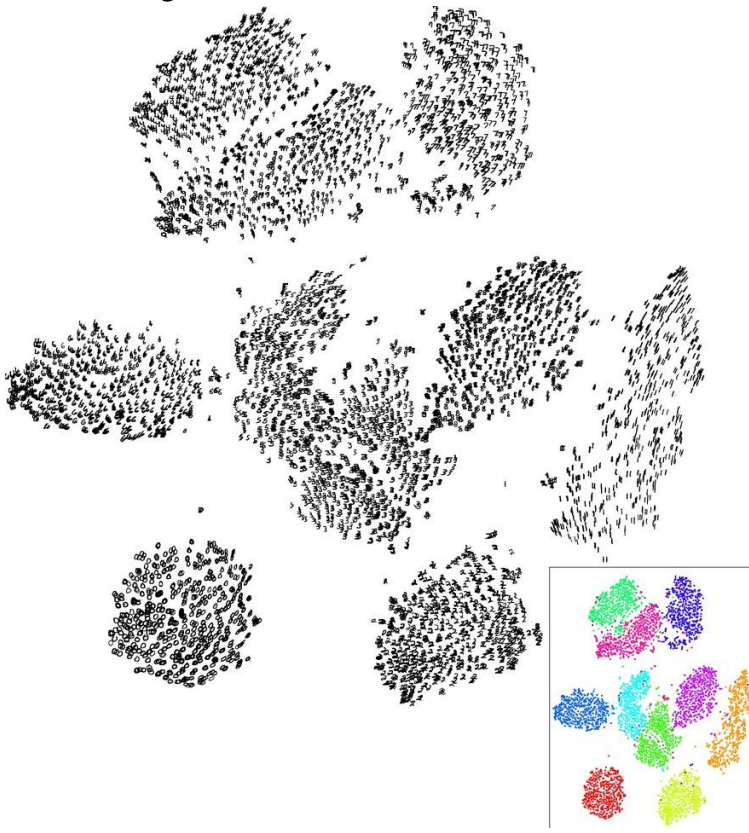
coactive.ai

Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

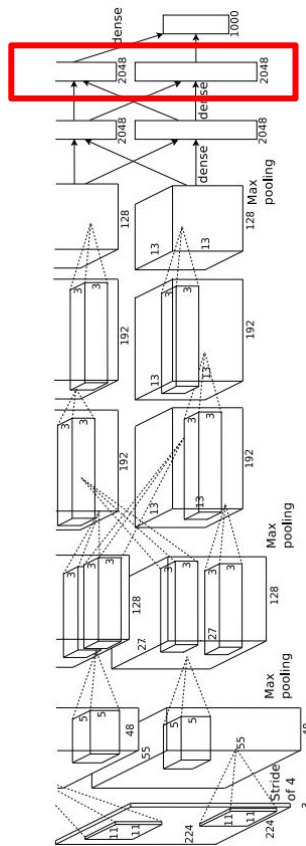
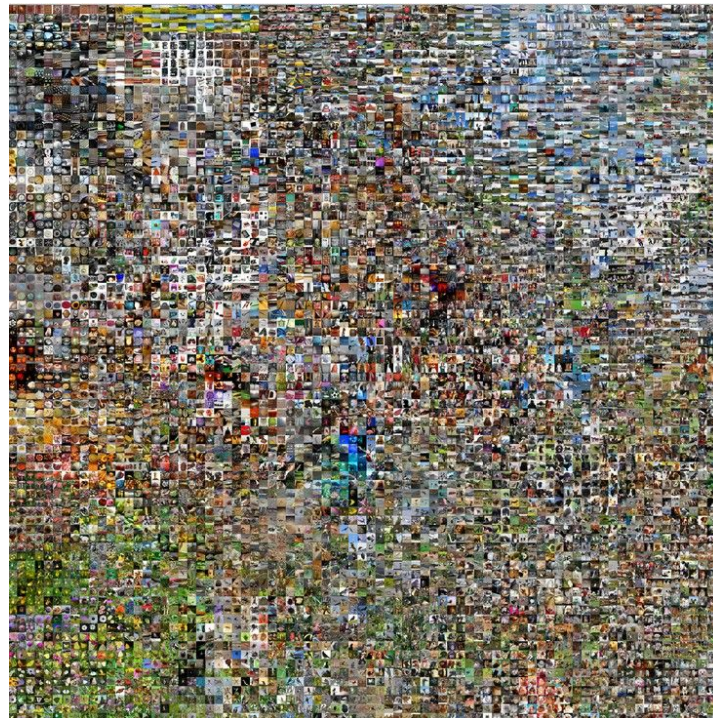
Simple algorithm: Principal Component Analysis (PCA)

More complex: **t-SNE**



Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.

Last Layer: Dimensionality Reduction



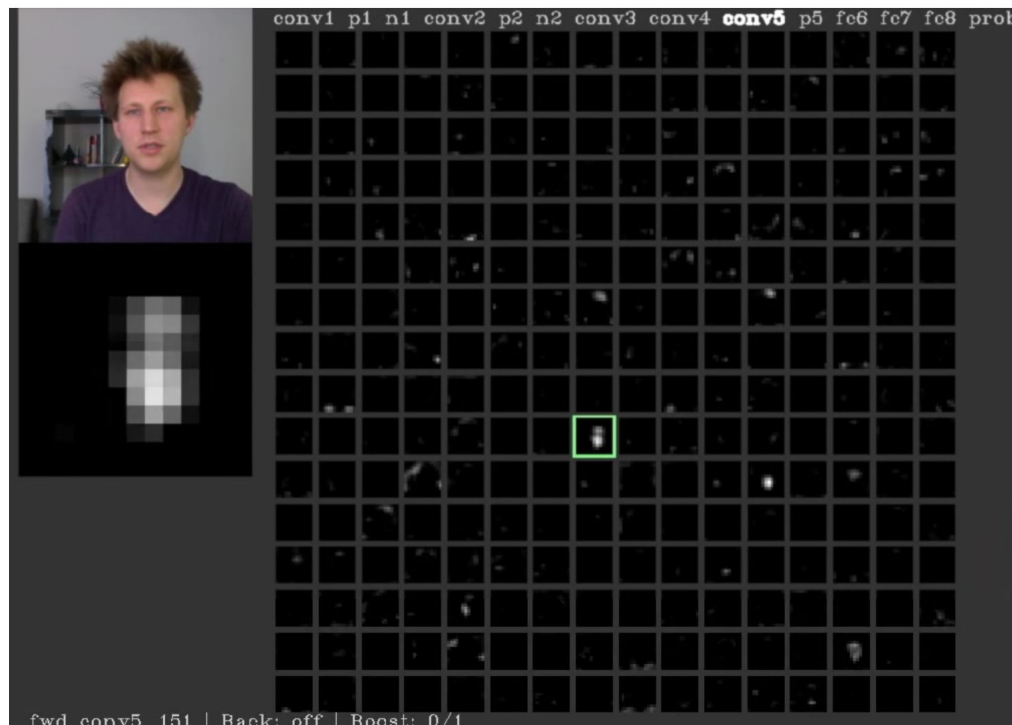
Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figure reproduced with permission.

See high-resolution versions at
<http://cs.stanford.edu/people/karpathy/cnnembed/>

Visualizing Activations

<https://www.youtube.com/watch?v=AgkflQ4IGaM>

conv5 feature map is
128x13x13; visualize
as 128 13x13
grayscale images



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, 2014. Reproduced with permission.

Today's agenda

Visualizing what models have learned:

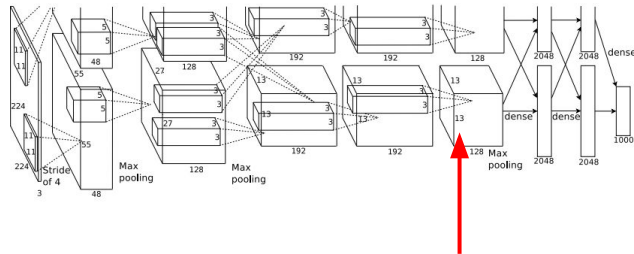
- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

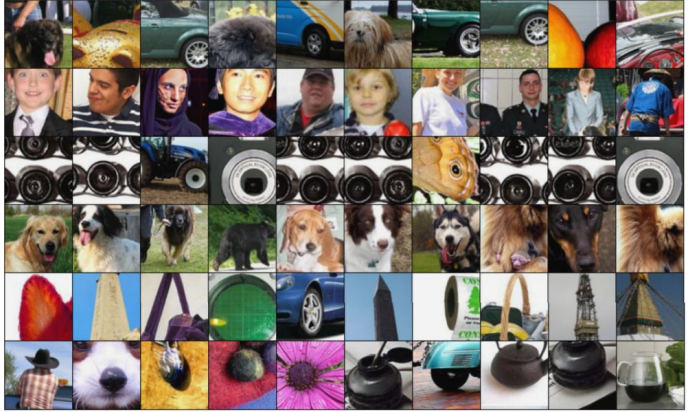
Maximally Activating Patches



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

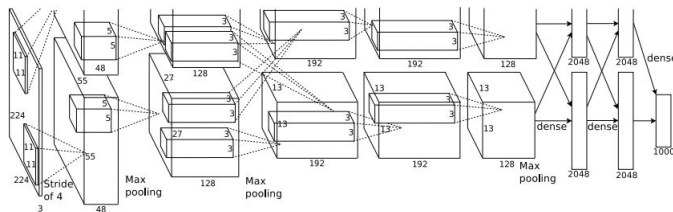
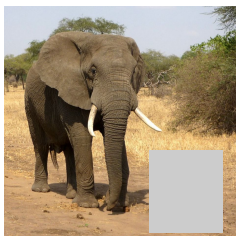
Visualize image patches that correspond to maximal activations



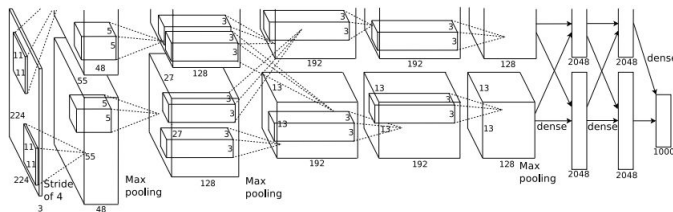
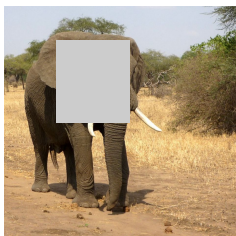
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
 Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Which pixels matter: Saliency via Occlusion

Mask part of the image before feeding to CNN,
check how much predicted probabilities change



$P(\text{elephant}) = 0.95$



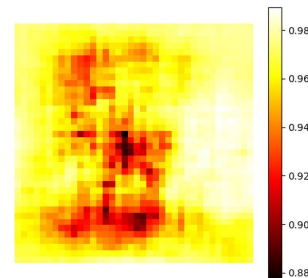
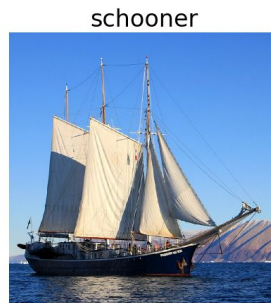
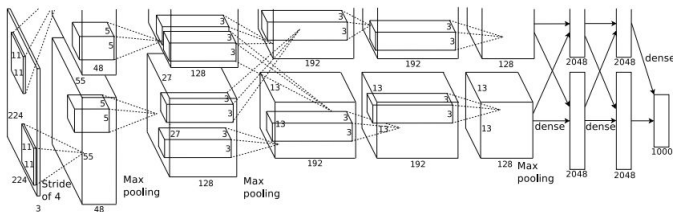
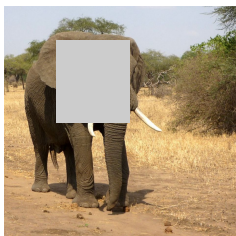
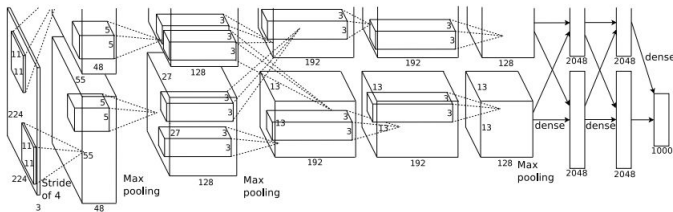
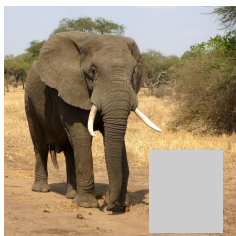
$P(\text{elephant}) = 0.75$

[Boat image](#) is [CC0 public domain](#)
[Elephant image](#) is [CC0 public domain](#)
[Go-Karts image](#) is [CC0 public domain](#)

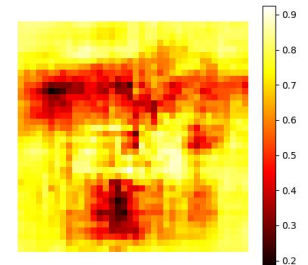
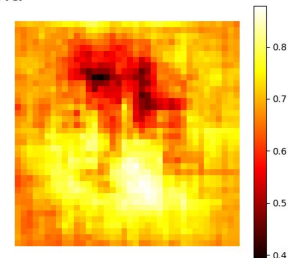
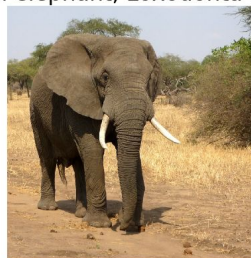
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Which pixels matter: Saliency via Occlusion

Mask part of the image before feeding to CNN,
check how much predicted probabilities change



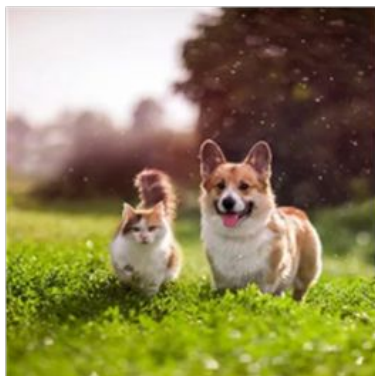
African elephant, *Loxodonta africana*



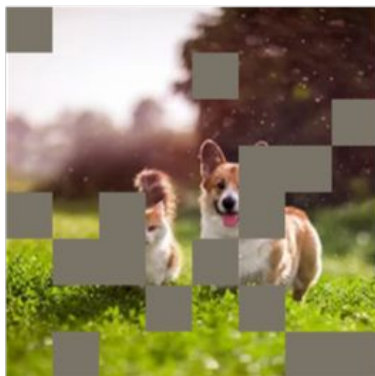
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

[Boat image](#) is [CC0 public domain](#)
[Elephant image](#) is [CC0 public domain](#)
[Go-Karts image](#) is [CC0 public domain](#)

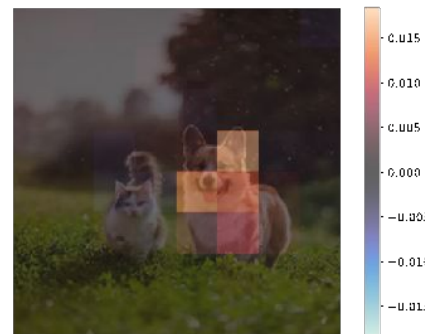
Saliency via Occlusion: Shapley Values



$P(\text{corgi}) = 0.99$



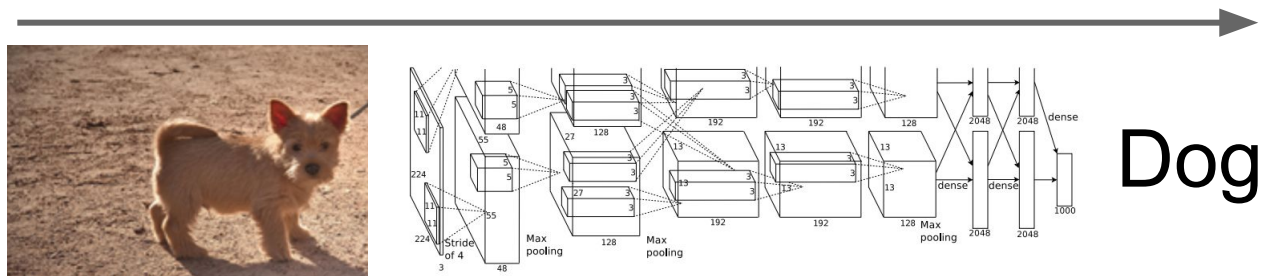
$P(\text{corgi}) = 0.8$



Credit: Ian Covert; Lundberg & Lee 2017

Which pixels matter: Saliency via Backprop

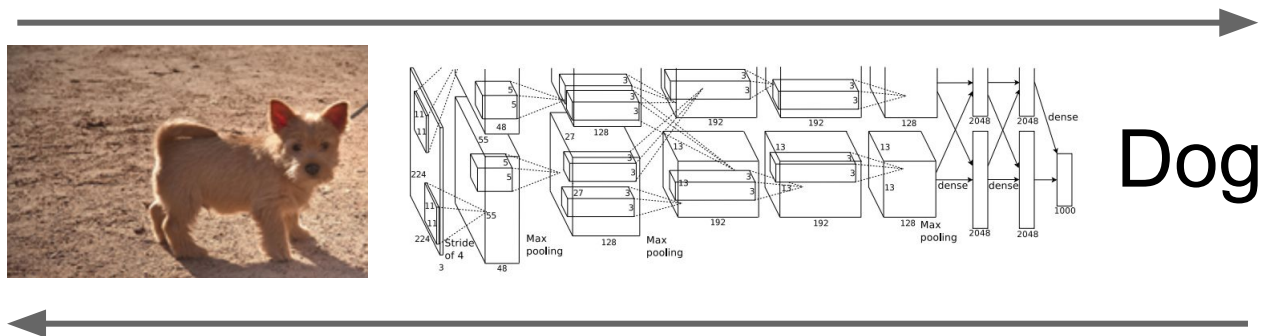
Forward pass: Compute probabilities



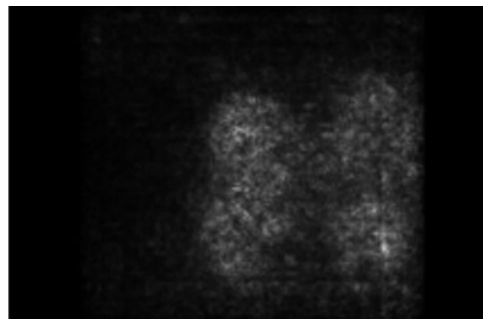
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

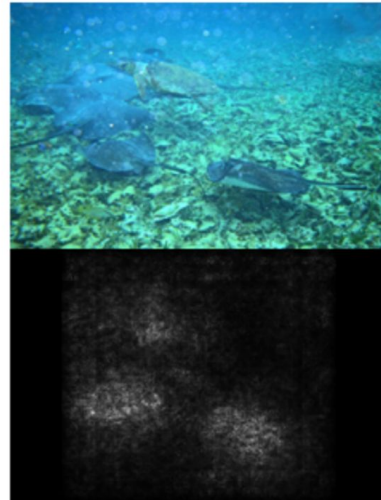
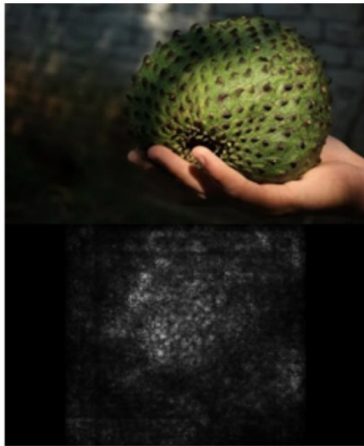
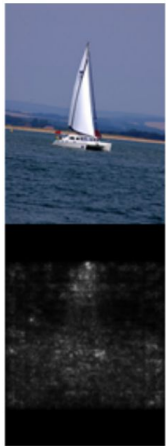


Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



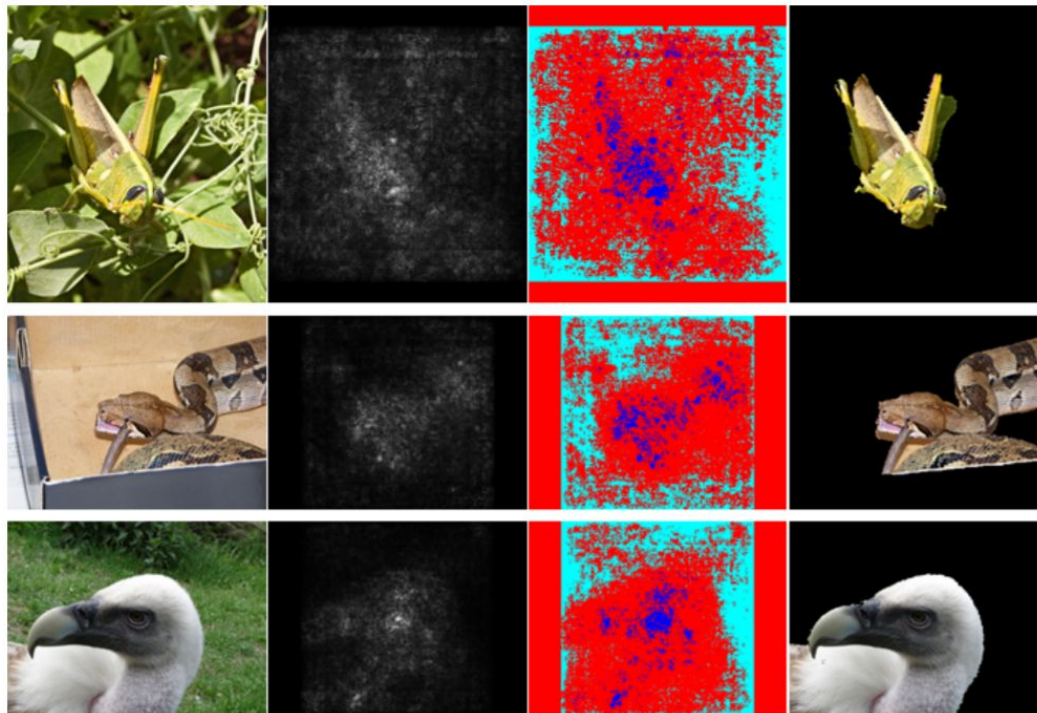
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Saliency Maps



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Saliency Maps: Segmentation without supervision



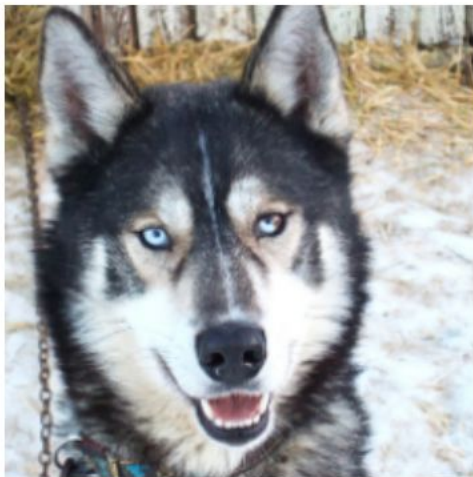
Use GrabCut on saliency map

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.
Rother et al, "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

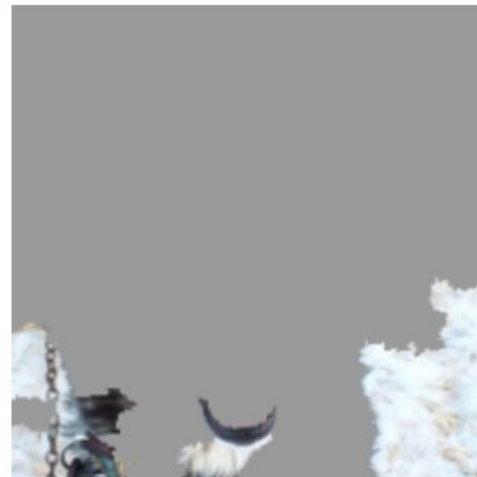
Saliency maps: Uncovers biases

Such methods also find biases

wolf vs dog classifier looks is actually a snow vs no-snow classifier



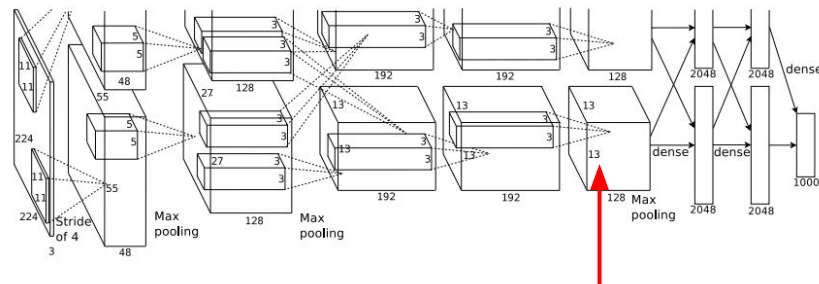
(a) Husky classified as wolf



(b) Explanation

Figures copyright Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, 2016; reproduced with permission. Ribeiro et al, "Why Should I Trust You?" Explaining the Predictions of Any Classifier", ACM KDD 2016

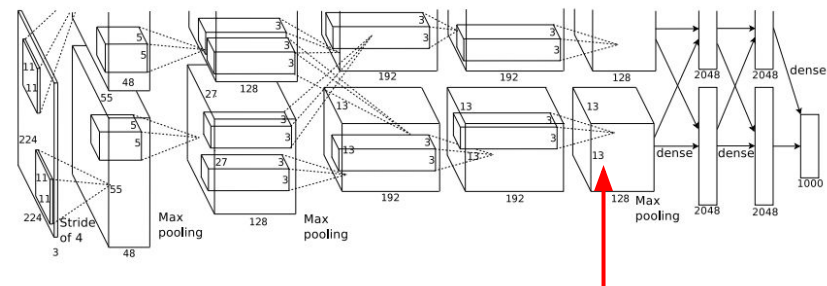
Intermediate Features via (guided) backprop



Pick a single intermediate channel, e.g. one value in 128 x 13 x 13 conv5 feature map

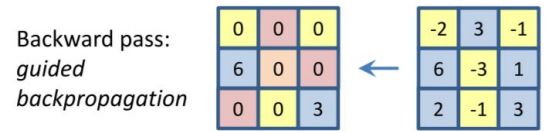
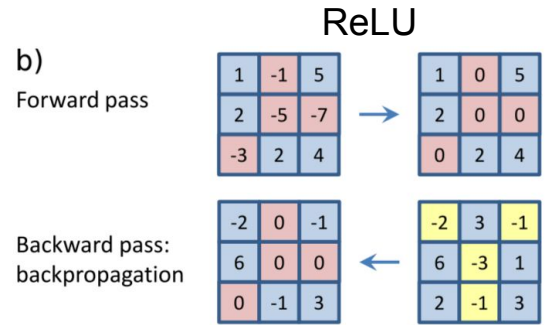
Compute gradient of activation value with respect to image pixels

Intermediate Features via (guided) backprop



Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels



Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Intermediate features via (guided) backprop



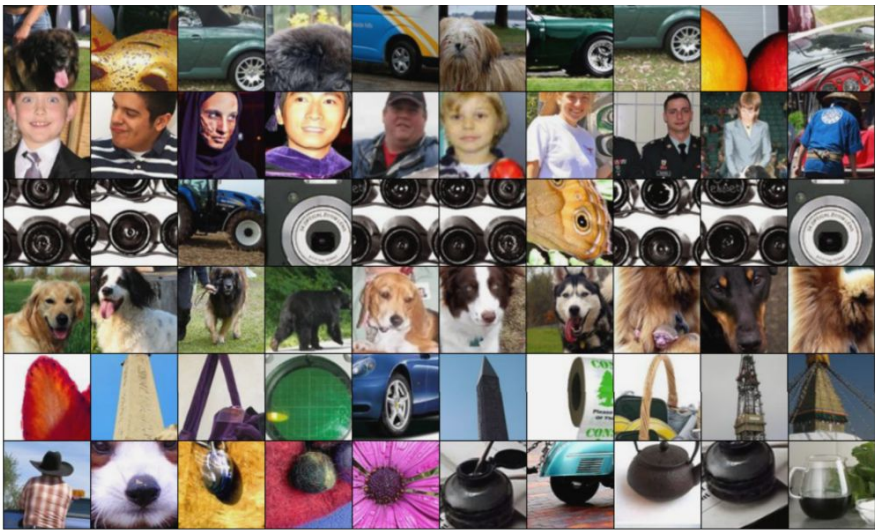
Maximally activating patches
(Each row is a different neuron)



Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Intermediate features via (guided) backprop



Maximally activating patches
(Each row is a different neuron)



Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Visualizing CNN features: Gradient Ascent

(Guided) backprop:

Find the part of an image that a neuron responds to

Gradient ascent:

Generate a synthetic image that maximally activates a neuron

$$I^* = \arg \max_I \boxed{f(I)} + \boxed{R(I)}$$

Neuron value

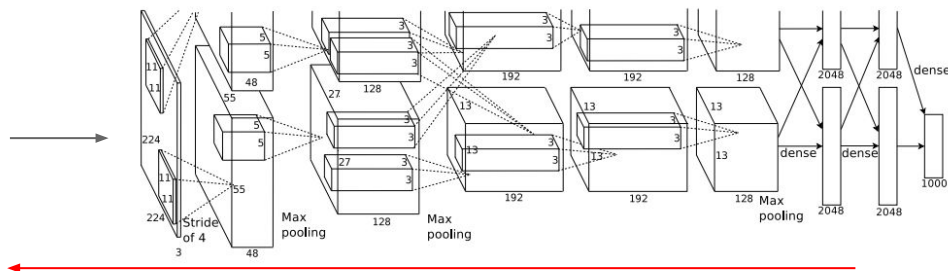
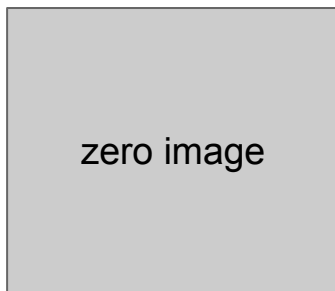
Natural image regularizer

Visualizing CNN features: Gradient Ascent

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

1. Initialize image to zeros



Repeat:

2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Visualizing CNN features: Gradient Ascent

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Simple regularizer: Penalize L2 norm of generated image

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Visualizing CNN features: Gradient Ascent

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Simple regularizer: Penalize L2 norm of generated image



dumbbell



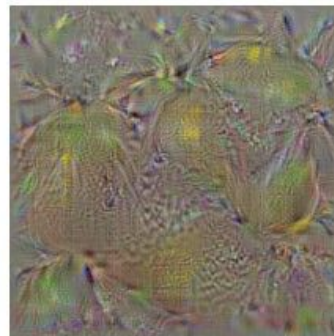
cup



dalmatian



bell pepper



lemon



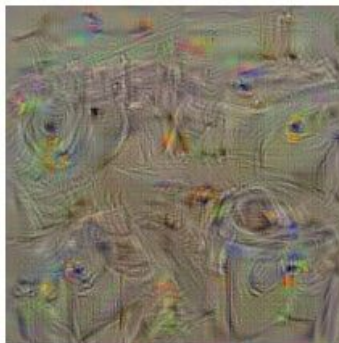
husky

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

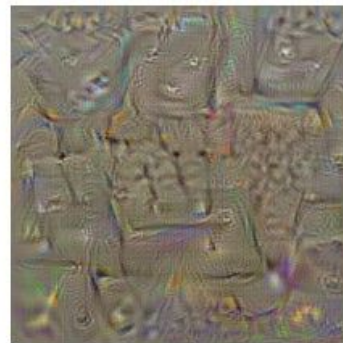
Visualizing CNN features: Gradient Ascent

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

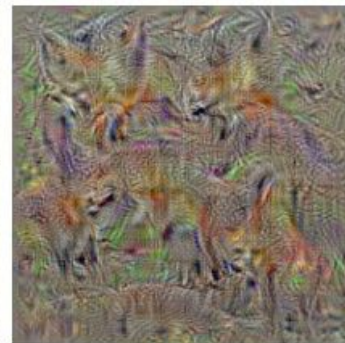
Simple regularizer: Penalize L2 norm of generated image



washing machine



computer keyboard



kit fox



goose



ostrich



limousine

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0

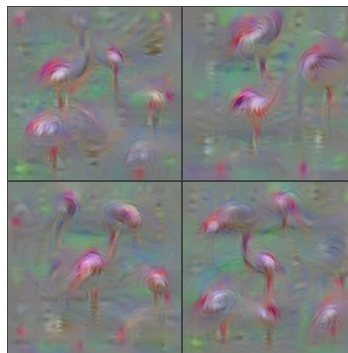
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualizing CNN features: Gradient Ascent

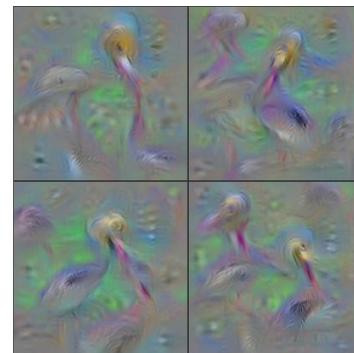
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

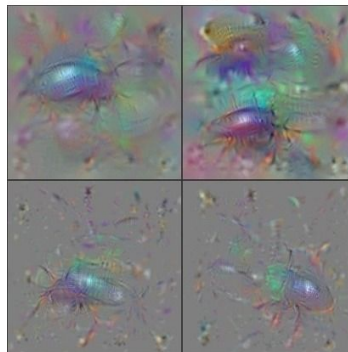
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



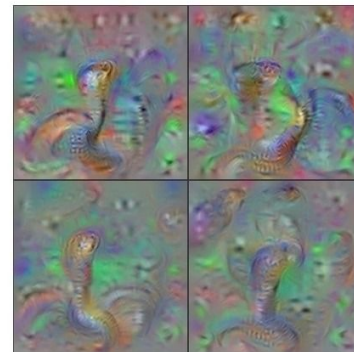
Flamingo



Pelican



Ground Beetle



Indian Cobra

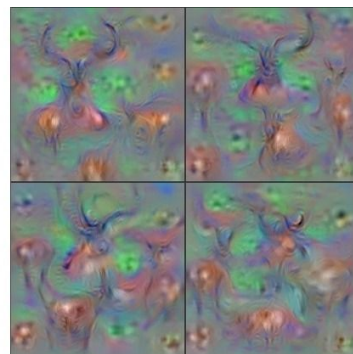
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

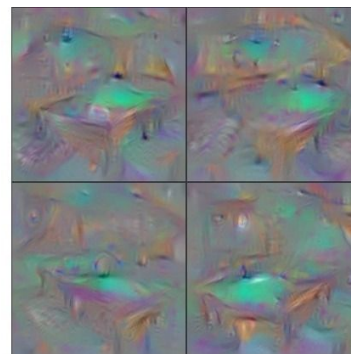
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

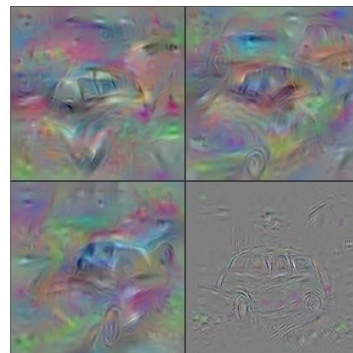
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



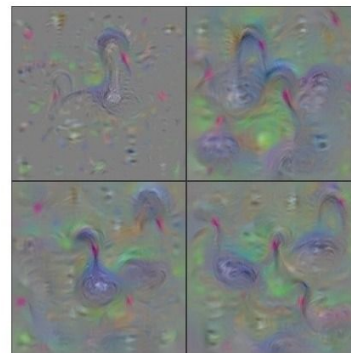
Hartebeest



Billiard Table



Station Wagon

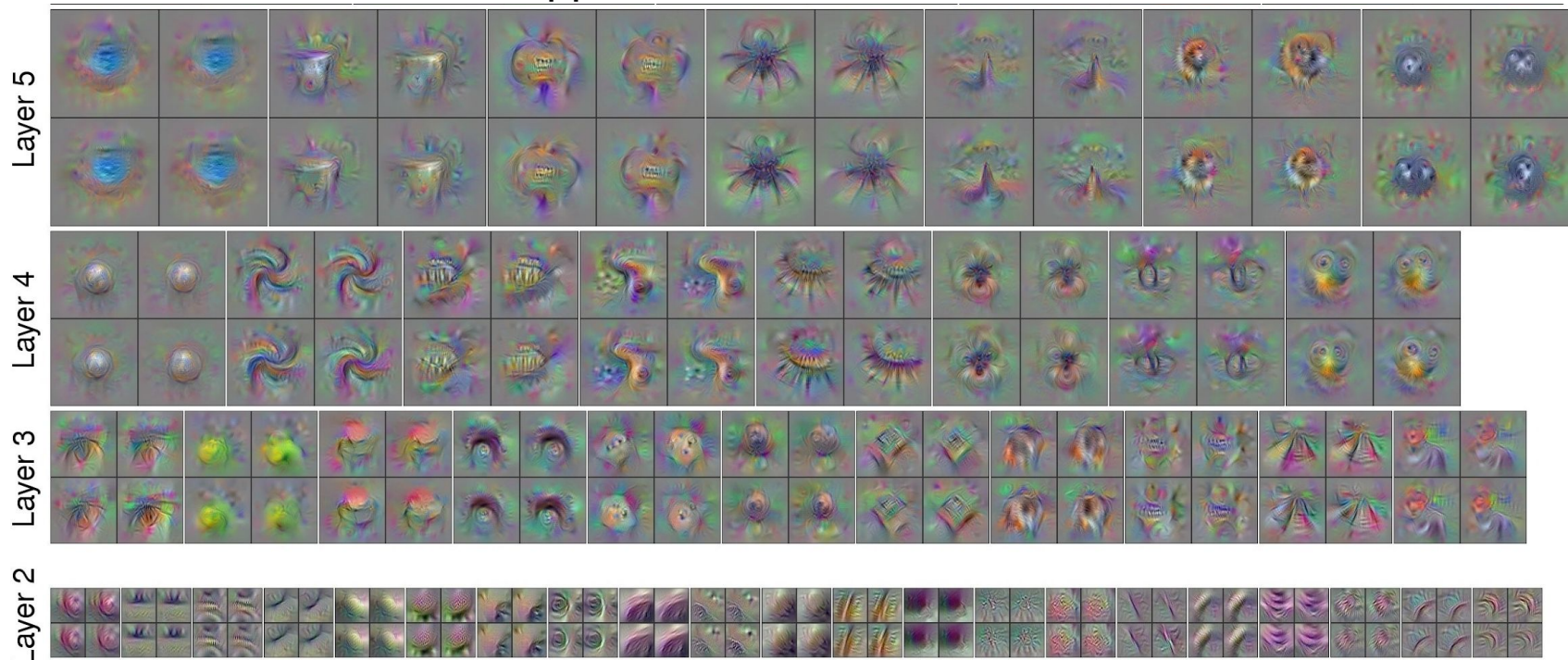


Black Swan

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

Use the same approach to visualize intermediate features



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

Adding “multi-faceted” visualization gives even nicer results:
(Plus more careful regularization, center-bias)

Reconstructions of multiple feature types (facets) recognized
by the same “grocery store” neuron



Corresponding example training set images recognized
by the same neuron as in the “grocery store” class



Nguyen et al, “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks”, ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

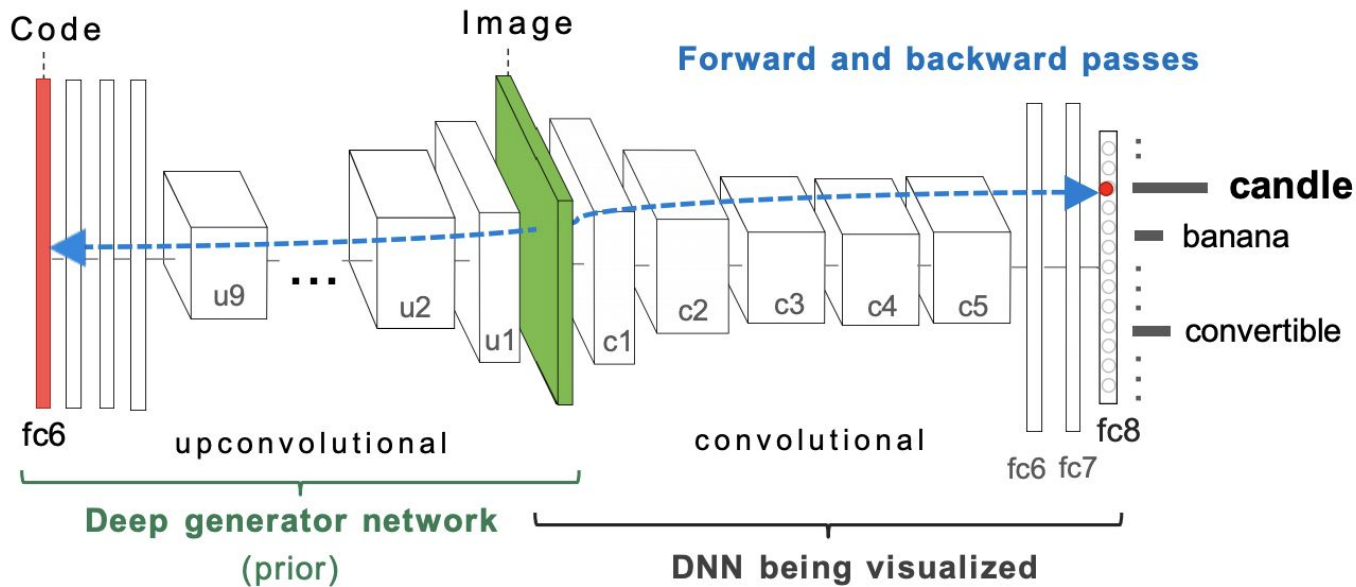
Visualizing CNN features: Gradient Ascent



Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned by Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

Visualizing CNN features: Gradient Ascent

Optimize in FC6 latent space instead of pixel space:



Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016
Figure copyright Nguyen et al, 2016; reproduced with permission.

Visualizing CNN features: Gradient Ascent

Optimize in FC6 latent space instead of pixel space:



Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016
Figure copyright Nguyen et al, 2016; reproduced with permission.

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

Fooling Images / Adversarial Examples

- (1) Start from an arbitrary image
- (2) Pick an arbitrary class
- (3) Modify the image to maximize the class
- (4) Repeat until network is fooled

Fooling Images / Adversarial Examples

African elephant



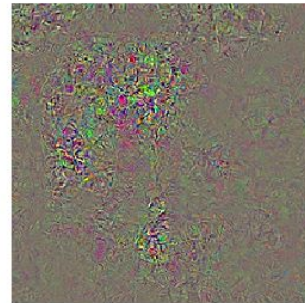
koala



Difference



10x Difference



schooner



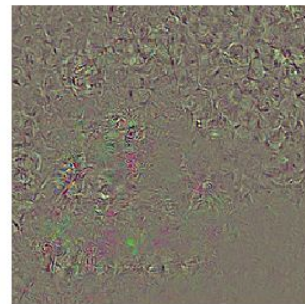
iPod



Difference



10x Difference



[Boat image](#) is [CC0 public domain](#)
[Elephant image](#) is [CC0 public domain](#)

Fooling Images / Adversarial Examples

African elephant



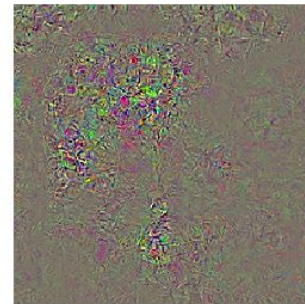
koala



Difference



10x Difference



schooner



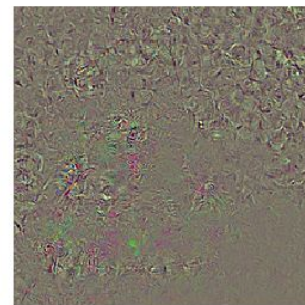
iPod



Difference



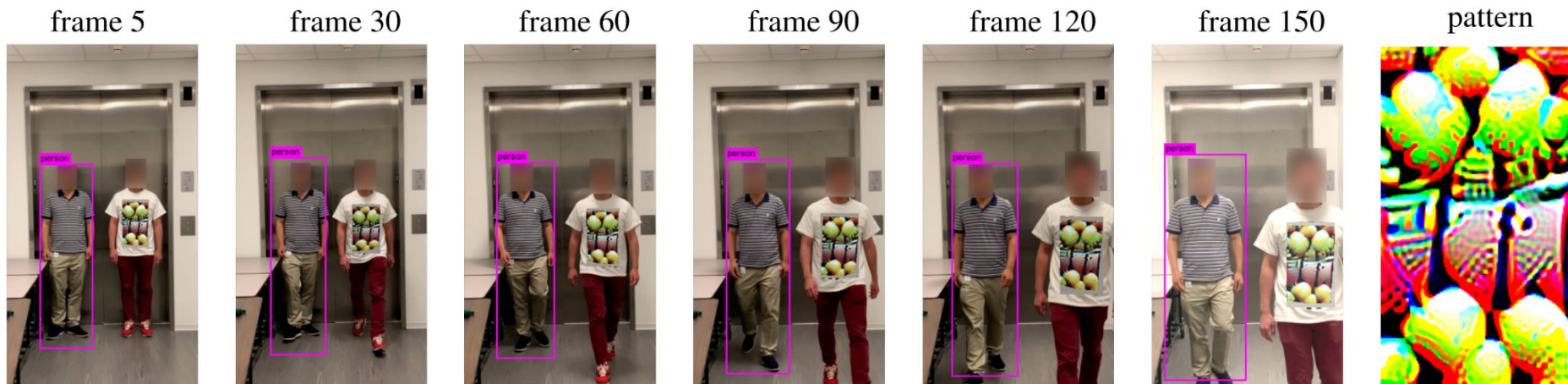
10x Difference



Boat image is [CC0 public domain](#)
Elephant image is [CC0 public domain](#)

Check out [Ian Goodfellow's lecture](#) from 2017

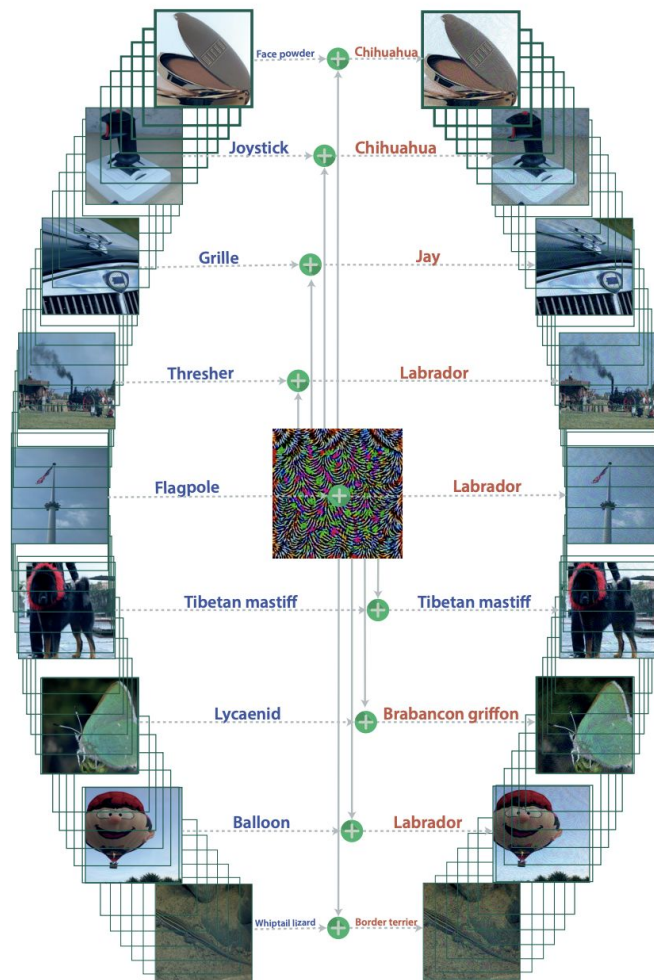
Fooling Person Detectors and Self-driving Cars



Xu et al., 2019; Eykholt et al., 2018

Fooling Images / Adversarial Example

Universal perturbations



Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
Figure reproduced with permission

Summary

Many methods for understanding CNN representations

Activations: Nearest neighbors, Dimensionality reduction, maximal patches, occlusion

Gradients: Saliency maps, class visualization, fooling images, feature inversion

Adversarial Examples: To confuse the models

Next time: **Introduction to language**