Convolutional Neural Networks
Regression and Keras Functional API

Alireza Akhavan Pour

AkhavanPour.ir
CLASS.VISION

Thursday, February 14, 2019

- ❑بخش ۱: روش‌های مختلف تعریف مدل
- ❑بخش ۲: چالش گربه / سگ در Kaggle
- ❑بخش ۳: افزایش دادگان
- ❑بخش ۴: بررسی معماری‌های حائز رتبه در طبقه‌بندی تصاویر
- ❑بخش ۵: انتقال یادگیری
- ❑بخش ۶: رگرسیون داده های ساختار یافته
- ❑بخش ۷: مدل با چند ورودی / چند خروجی

چرا کانولوشن؟

$$76\ x\ 6\ =\ 456(5\ x\ 5\ x\ 3)\ +\ 1\ =76$$

$$3072\ \times 4704 = 14450688$$
$$\approx\ 14.5\ m$$

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

بخش ۱:

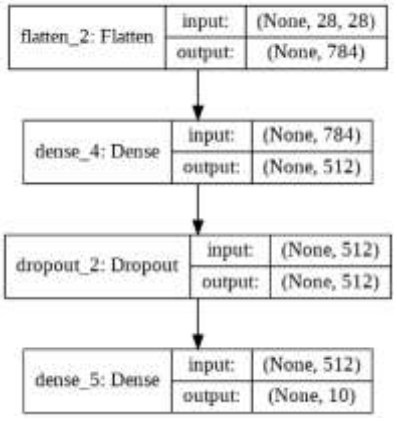
روش‌های مختلف تعریف مدل
Symbolic and Imperative APIs in TensorFlow 2.0

What are Symbolic and Imperative APIs in TensorFlow 2.0?

- ❑Sequential model
- ❑Functional API
- ❑Model subclassing

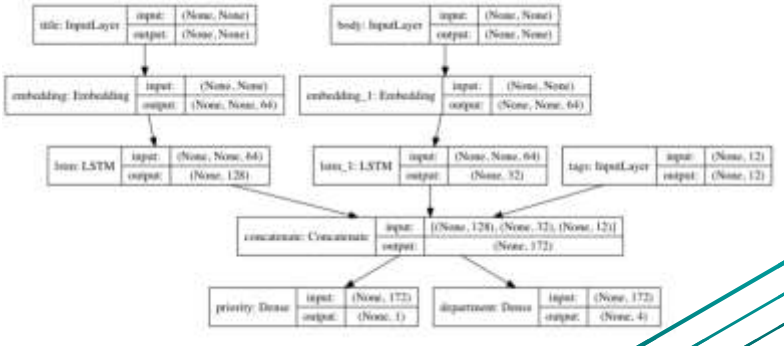
What are Symbolic and Imperative APIs in TensorFlow 2.0?

- ❑**Sequential model**
- ❑Functional API
- ❑Model subclassing



What are Symbolic and Imperative APIs in TensorFlow 2.0?

- ❑Sequential model
- ❑**Functional API**
- ❑Model subclassing



What are Symbolic and Imperative APIs in TensorFlow 2.0?

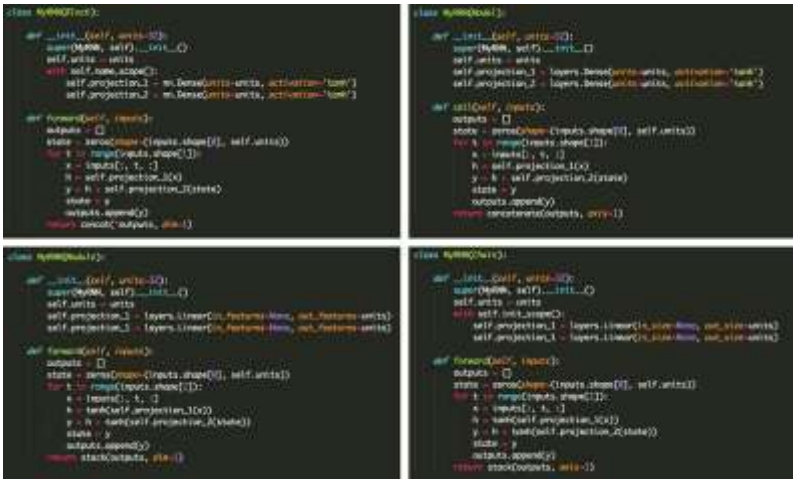
- ❑Sequential model
- ❑Functional API
- ❑**Model subclassing**

```
class CNN_Encoder(tf.keras.Model):
  def __init__(self, embedding_dim):
    super(CNN_Encoder, self).__init__()
    self.fc = tf.keras.layers.Dense(embedding_dim)

  def call(self, x):
    x = self.fc(x)
    x = tf.nn.relu(x)
    return x
```

https://medium.com/tensorflow/what-are-symbolic-and-imperative-apis-in-tensorflow-2-0-dfccecb01021
https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/guide/keras.ipynb

TensorFlow/Keras, MXNet/Gluon, Chainer, PyTorch



https://twitter.com/fchollet/status/1052228463300493 3

بخش ۲:

پیاده سازی چالش گربه/سگ
Cat VS Dog

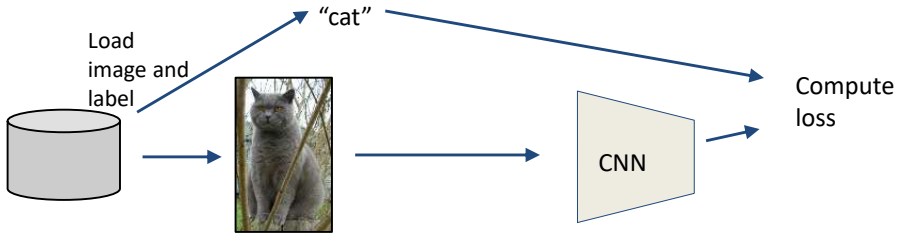
شبکه‌های کانولوشنالی در کراس – چالش گربه / سگ

07_CNN-cat_Vs_dog.ipynb

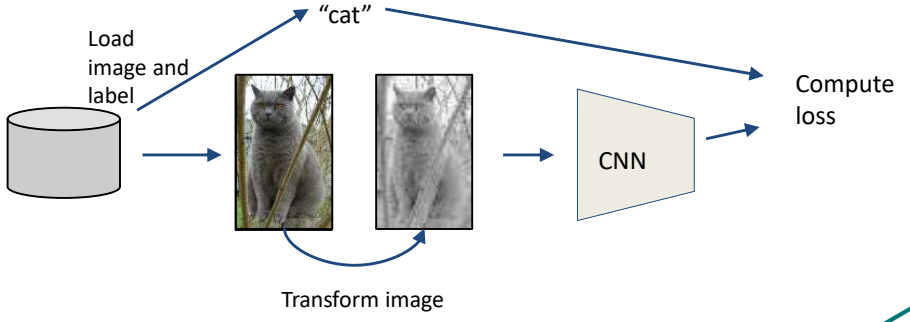
بخش ۳:

افزایش دادگان
Data Augmentation

Data Augmentation

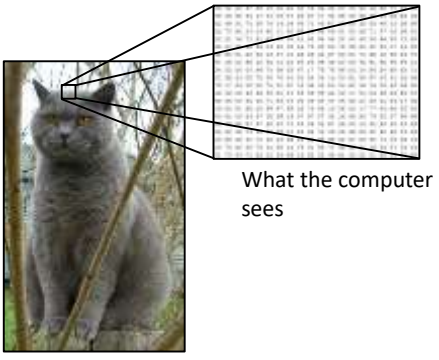


Data Augmentation



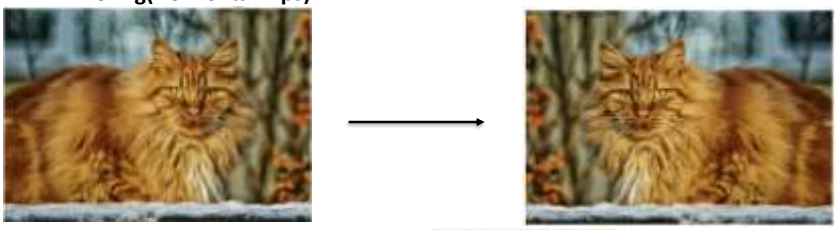
Data Augmentation

- Change the pixels without changing the label
- Train on transformed data
- VERY widely used

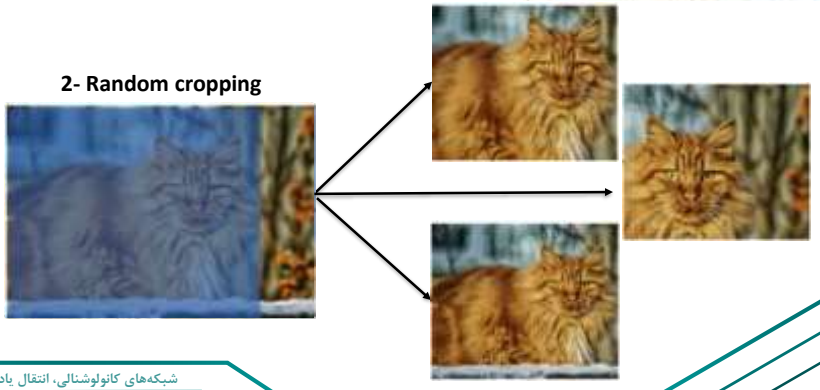


Common augmentation method

1- Mirroring(Horizontal flips)

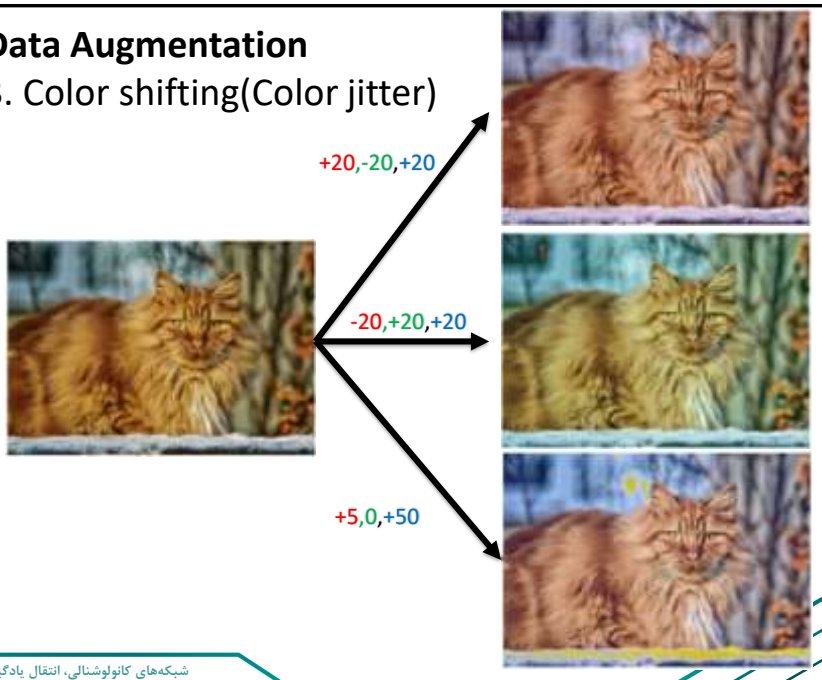


2- Random cropping



Data Augmentation

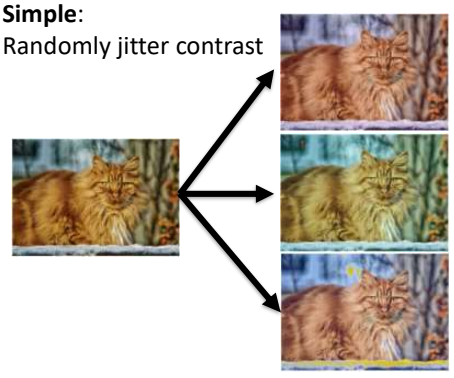
3. Color shifting(Color jitter)



Data Augmentation

3. Color shifting(Color jitter)

Simple:
Randomly jitter contrast



برای مطالعه بیشتر ☺

Complex (PCA color augmentation):

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a "color offset" along principal component directions

(As seen in *[Krizhevsky et al. 2012],* ResNet, etc)

Data Augmentation

4. Get creative!

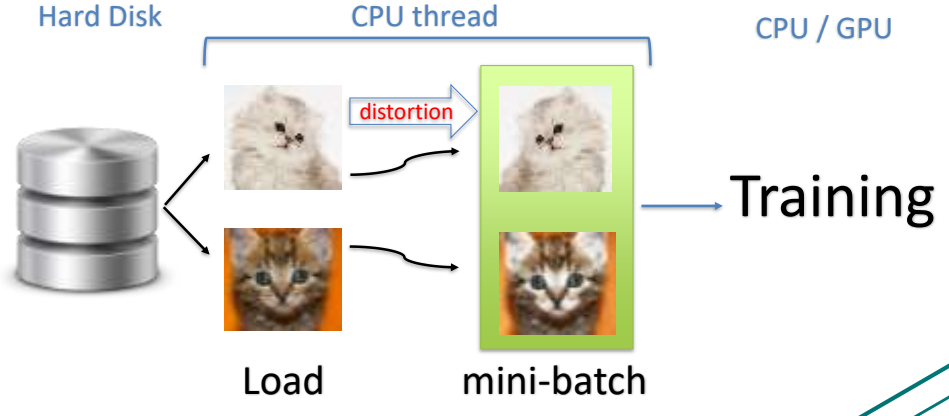
Random mix/combinations of :

- translation
- rotation
- stretching
- shearing, 
- lens distortions, … (go crazy)

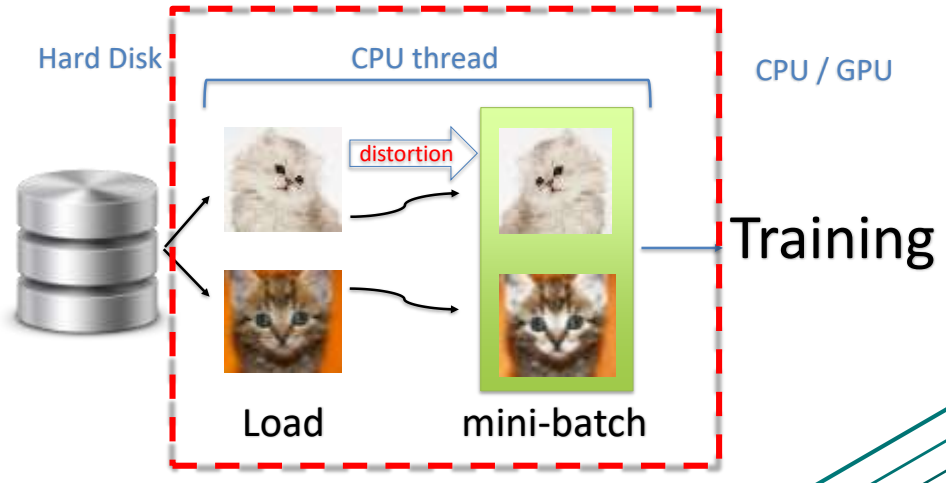
Bottlenecks

to be aware of

Implementation distortions during training



Implementation distortions during training



Data Augmentation

08_data_augmentation.ipynb

لود مدل ذخیره شده در کراس

09_Load_trained_model_in_keras.ipynb

بخش ۴:

مطالعه موردی

(بررسی معماری‌های حائز رتبه در طبقه بندی تصاویر)

Case Study

Classic networks:

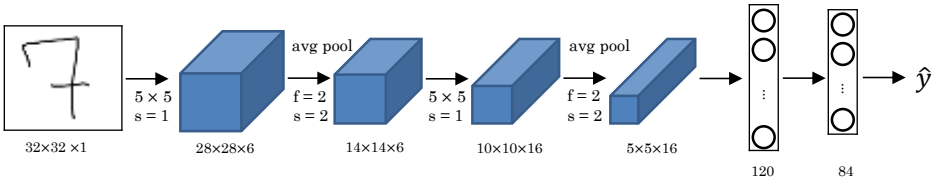
- LeNet-5
- AlexNet
- ZFNet
- VGG

ResNet

Inception

LeNet

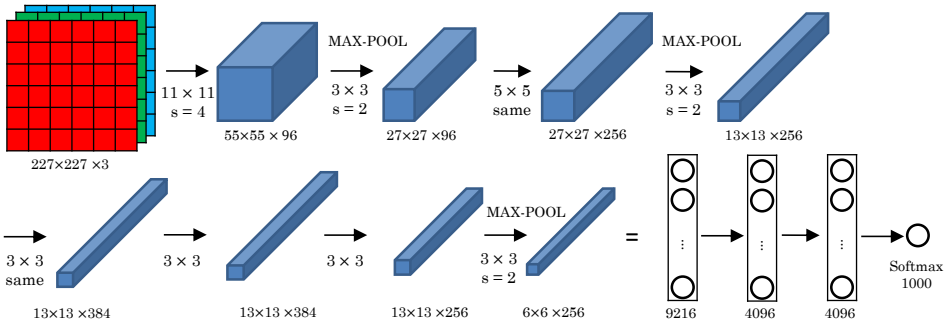
LeNet - 5



[LeCun et al., 1998. Gradient-based learning applied to document recognition]

AlexNet

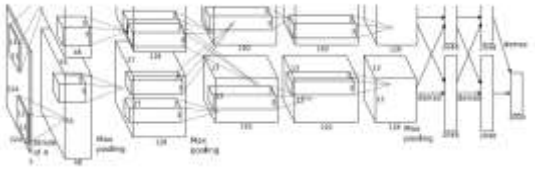
AlexNet



[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

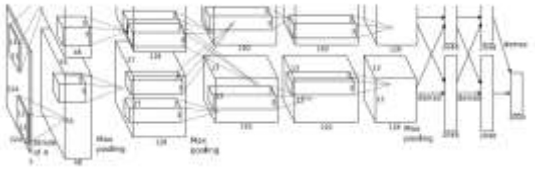
First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

Data augmentation، شبکه‌های کانولوشنالی، انتقال یادگیری
علیرضا اخوان پور

Case Study: AlexNet

[Krizhevsky et al. 2012]

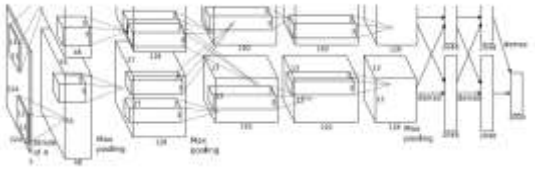


Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

Case Study: AlexNet

[Krizhevsky et al. 2012]



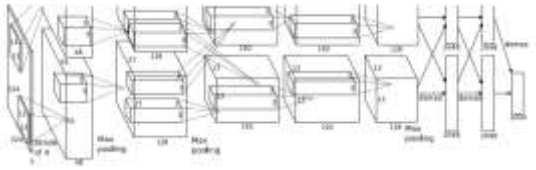
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

Case Study: AlexNet

[Krizhevsky et al. 2012]



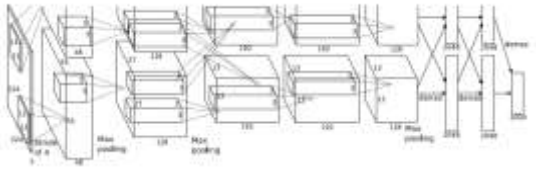
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

Case Study: AlexNet

[Krizhevsky et al. 2012]

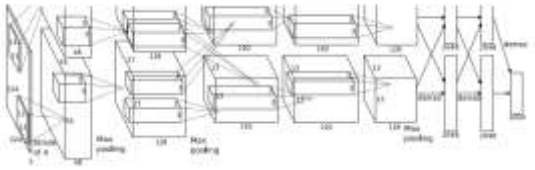


Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

Case Study: AlexNet

[Krizhevsky et al. 2012]

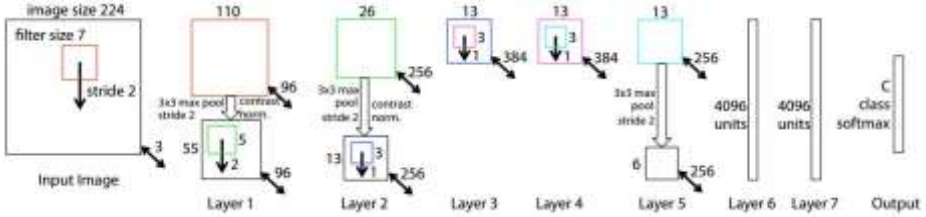


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- **7 CNN ensemble: 18.2% -> 15.4%**

ZFNet

Case Study: ZFNet



AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

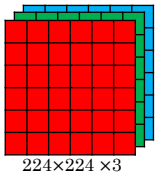
ImageNet top 5 error: 15.4% -> 14.8%

[Zeiler and Fergus, 2013]

VGG

VGG - 16

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2



224×224 ×3

→ [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] × 3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

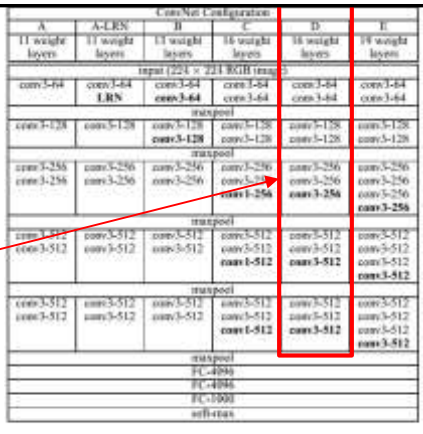
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

TOTAL memory: 24M * 4 bytes ~= 93MB / image
(only forward! ~*2 for bwd)
TOTAL params: 138M parameters

best model



ConvNet Configuration

A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A, A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

11.2% top 5 error in ILSVRC 2013 -> 7.3% top 5 error

Classic networks:

- LeNet-5 ✓
- AlexNet ✓ **7 CNN ensemble: 15.4%** top 5 error
- ZFNet ✓ 14.8% top 5 error
- VGG ✓ 7.3% top 5 error

Inception

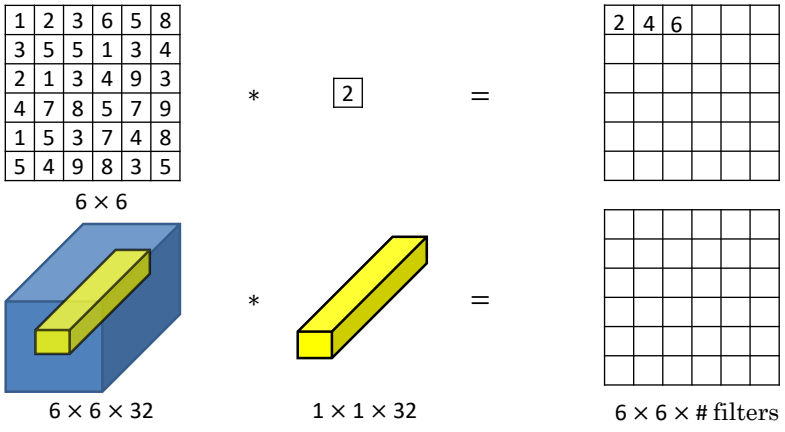
ResNet

اطلاعات بیشتر:
جلسه ۱۶ - بررسی معماری‌های حائز رتبه در طبقه بندی تصاویر:
http://blog.class.vision/winter-96-97-syllabus/

One by One Convolution!

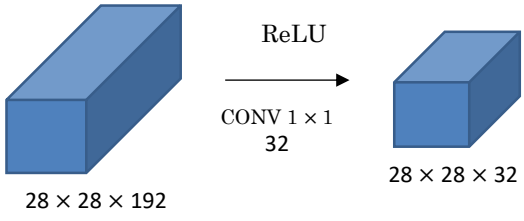
45

Why does a 1 × 1 convolution do?



[Lin et al., 2013. Network in network]

46



[Lin et al., 2013. Network in network]

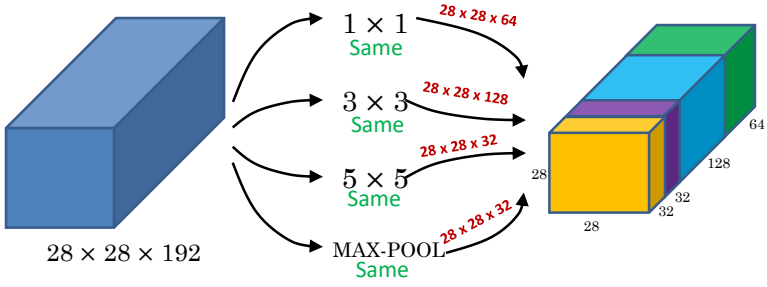
47

Inception (GoogLeNet)

48

http://knowyourmeme.com/memes/we-need-to-go-deeper

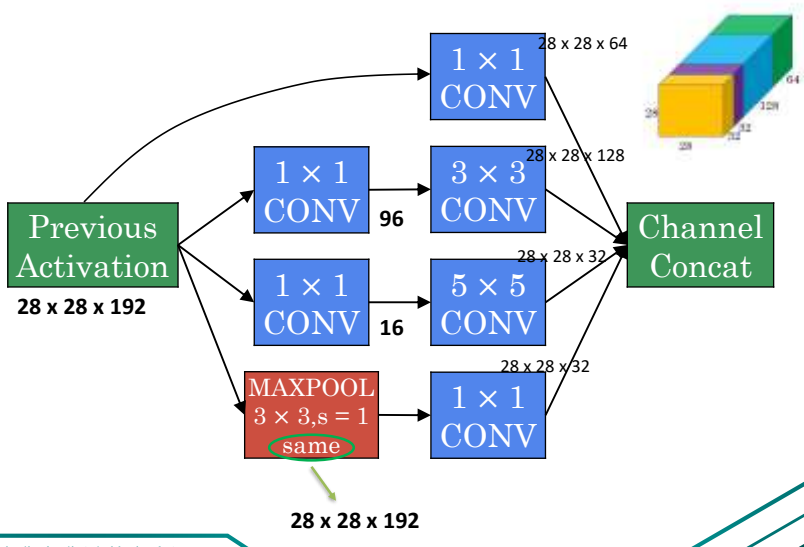
49



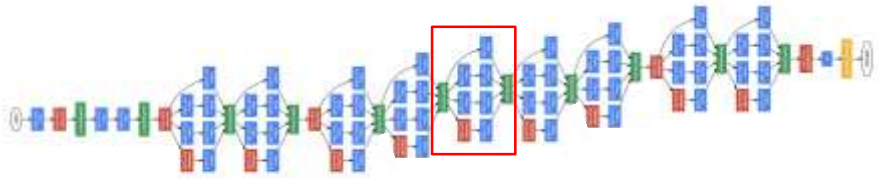
[Szegedy et al. 2014, Going deeper with convolutions]

50

Inception module



Inception network



[Szegedy et al., 2014, Going Deeper with Convolutions]

52

Case Study: GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

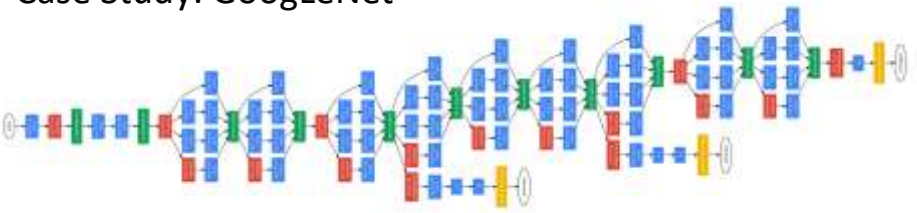
Fun features:

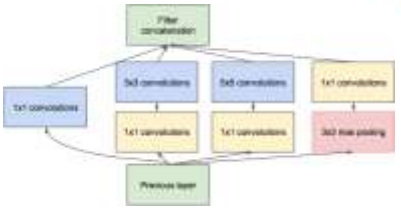
- Only 5 million params! (Removes FC layers completely)

Compared to AlexNet:
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

53

Case Study: GoogLeNet





Inception module

[Szegedy et al., 2014, Going Deeper with Convolutions]

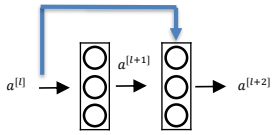
ILSVRC 2014 winner (6.7% top 5 error)

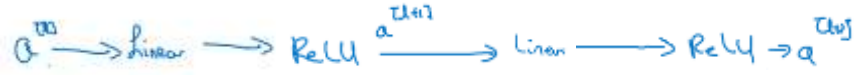
54

ResNet

55

Residual block





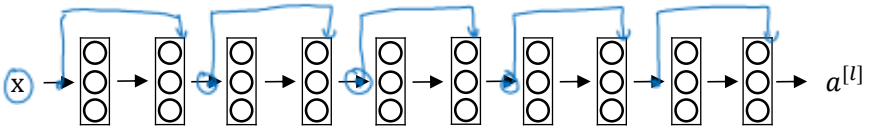
$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$

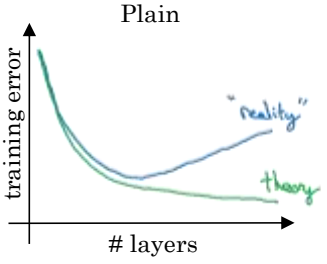
$$a^{[l+2]} = g\left(z^{[l+2]} + a^{[l]}\right)$$

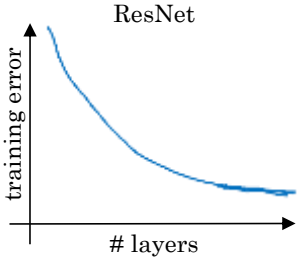
[He et al., 2015. Deep residual networks for image recognition]

56

Residual Network







[He et al., 2015. Deep residual networks for image recognition]

57

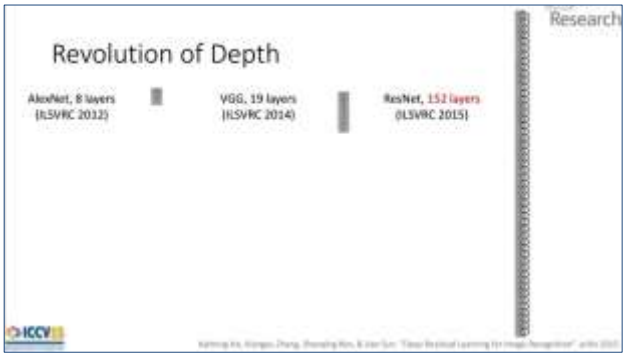
Case Study: ResNet *[He et al., 2015]*

Slide from Kaiming He's recent presentation
https://www.youtube.com/watch?v=1PGLj-uKT1w

58

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training on 8 GPU machine

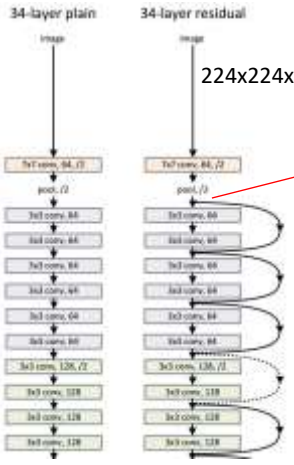
at runtime: faster than a VGGNet! (even though it has 8x more layers)

(slide from Kaiming He's recent presentation)

59

Case Study: ResNet

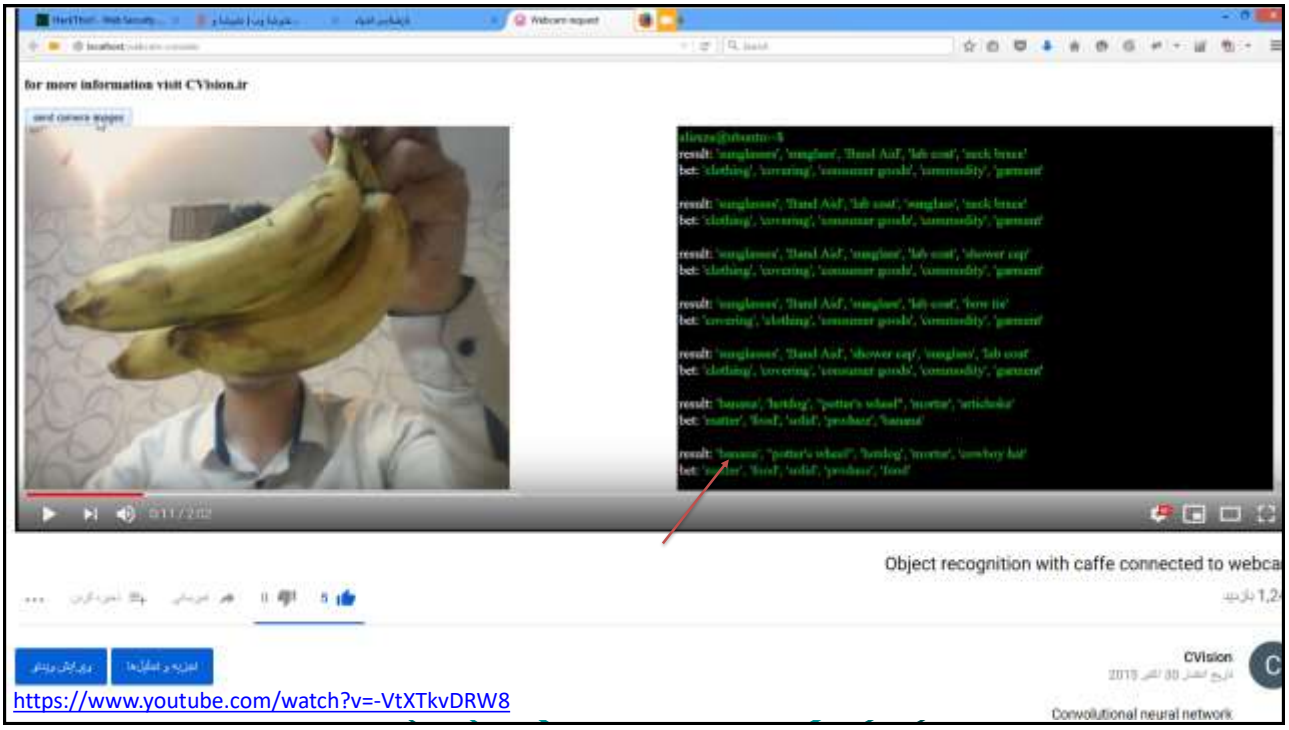
[He et al., 2015]



spatial dimension only
56x56!

استفاده از مدل‌های از قبل آموزش داده شده در کراس

10_using-a-pretrained-convnet.ipynb



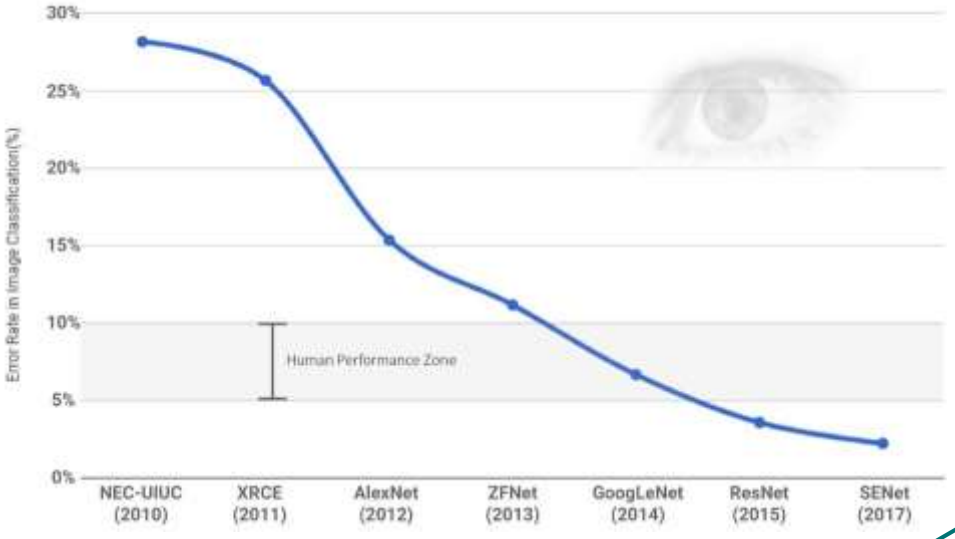
https://www.youtube.com/watch?v=-VtXTkvDRW8

استفاده از مدل‌های از قبل آموزش داده شده در کراس

11_using-a-pretrained-convnet-webcam.ipynb

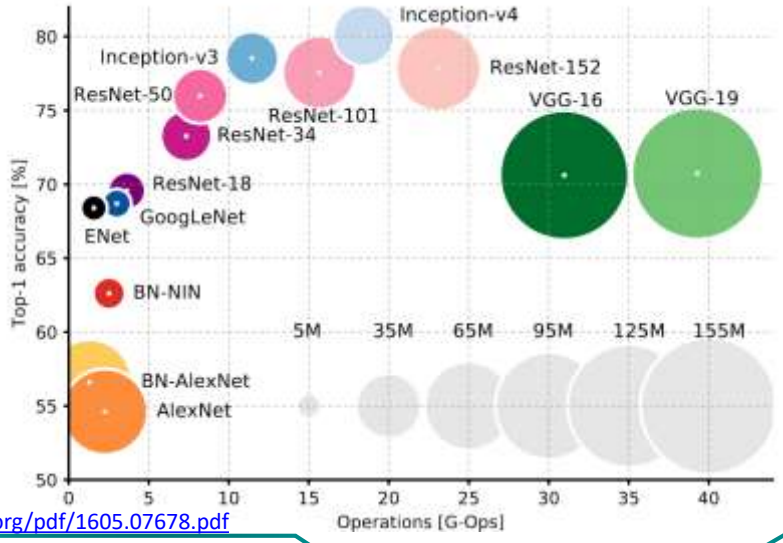
بخش ۵:

انتقال یادگیری
Transfer Learning



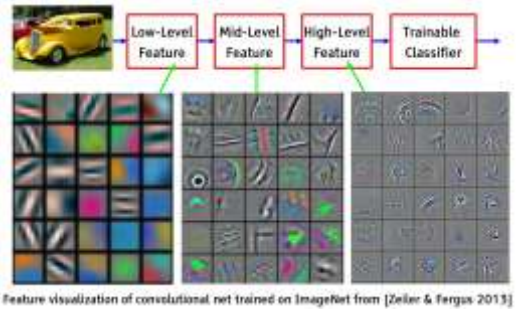
Data augmentation ،شبکه‌های کانولوشنالی، انتقال یادگیری
علیرضا اخوان پور

Accuracy vs Operations per image inference

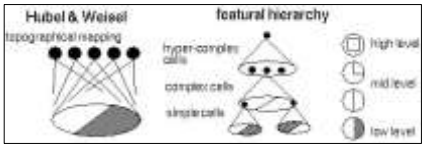


https://arxiv.org/pdf/1605.07678.pdf

Preview

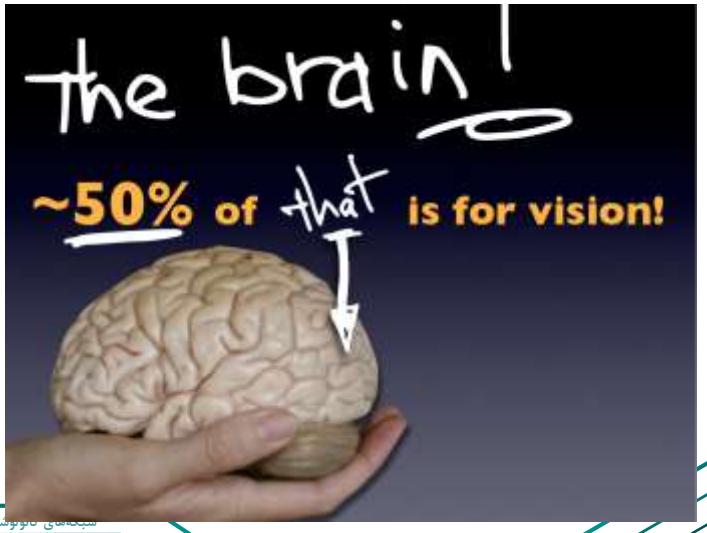


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



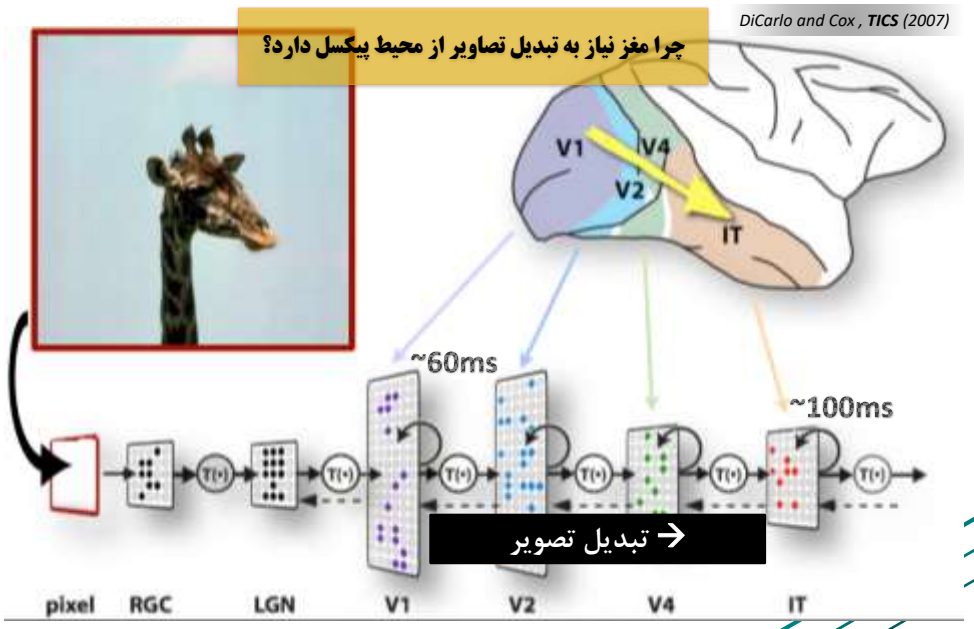
[From recent Yann LeCun slides]

نگاهی به فرآیند بینایی در مغز!



چرا مغز نیاز به تبدیل تصاویر از محیط پیکسل دارد؟

DiCarlo and Cox , **TICS** (2007)



تغییرات اشیاء

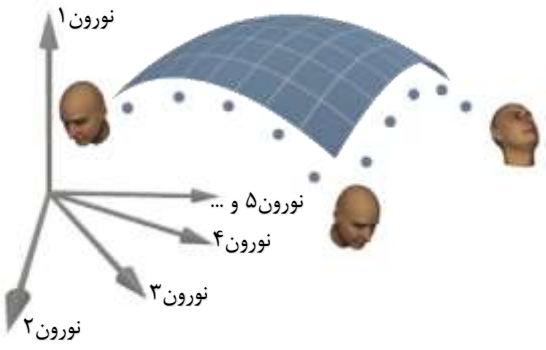
اندازه، زاویه‌ی دید، تغییرات نور و موقعیت در صفحه

درهم ریختگی (Clutter, occlusion)

Invariant object recognition

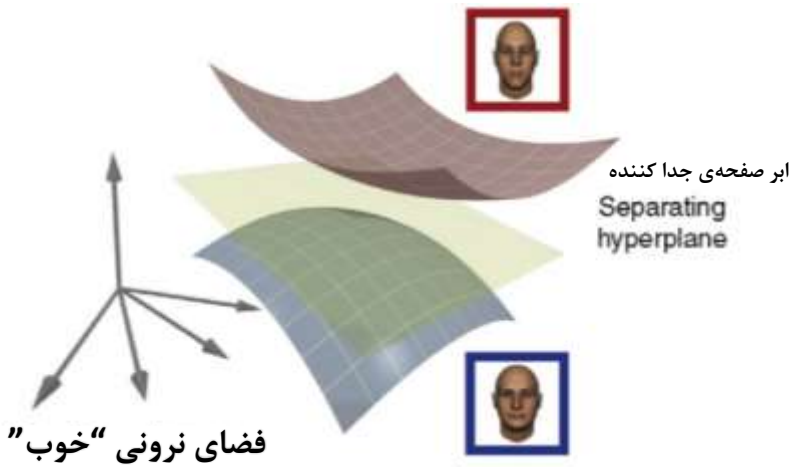
درون دسته‌ای (intraclass)

فضای نرونی و خمینه (manifold)

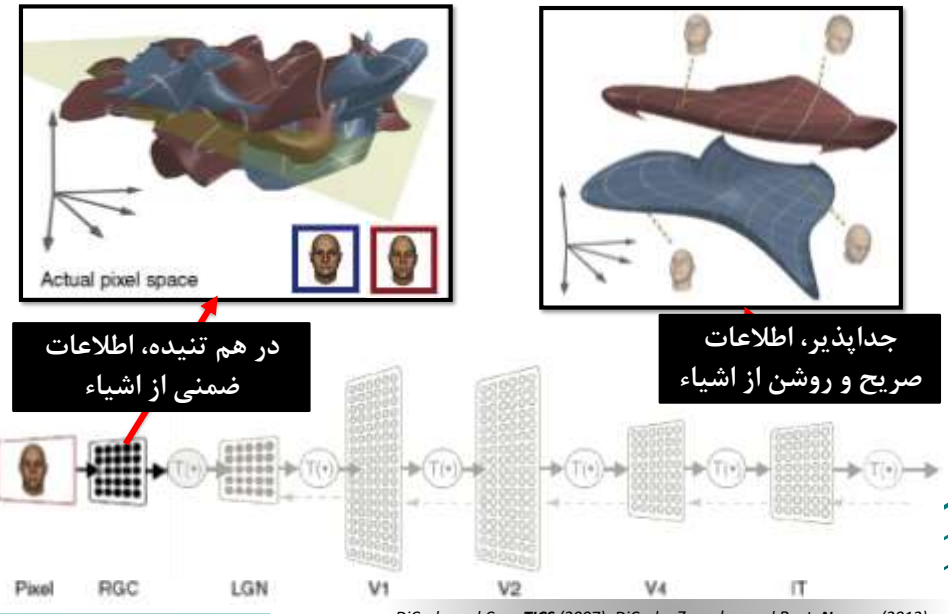


DiCarlo and Cox , ***TICS*** *(2007)*

فضای نرونی خوب



تبدیل فضای نورونی در مغز



*DiCarlo and Cox , **TICS** (2007), DiCarlo, Zoccolan and Rust, **Neuron** (2012)*

استخراج ویژگی با شبکه های عمیق از پیش آموزش داده شده

12_pretrained_convnet_feature_extraction-1.ipynb

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

Transfer Learning

"You need a lot of a dat[illegible] if you want to train/use CNNs"

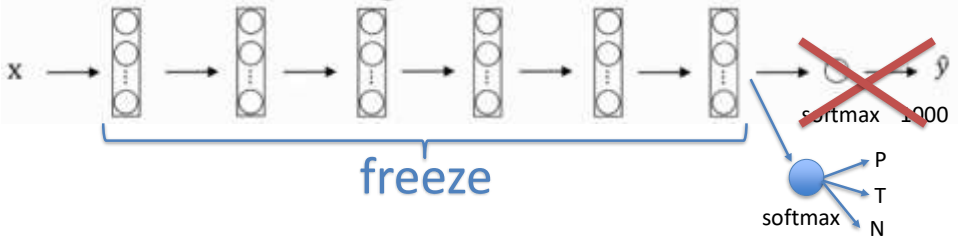
BUSTED

Transfer Learning


Persian Cat


Tiger cat

?
Neither



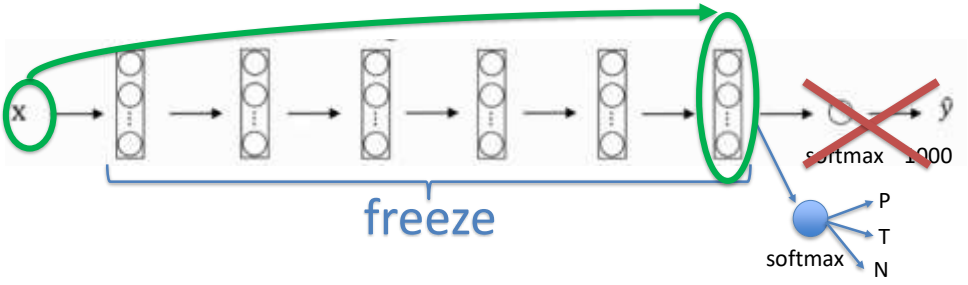
Transfer Learning


Persian Cat


Tiger cat

?

Neither



- میتوان از قبل خروجی لایه را برای تصاویر آموزشی حساب کرد.

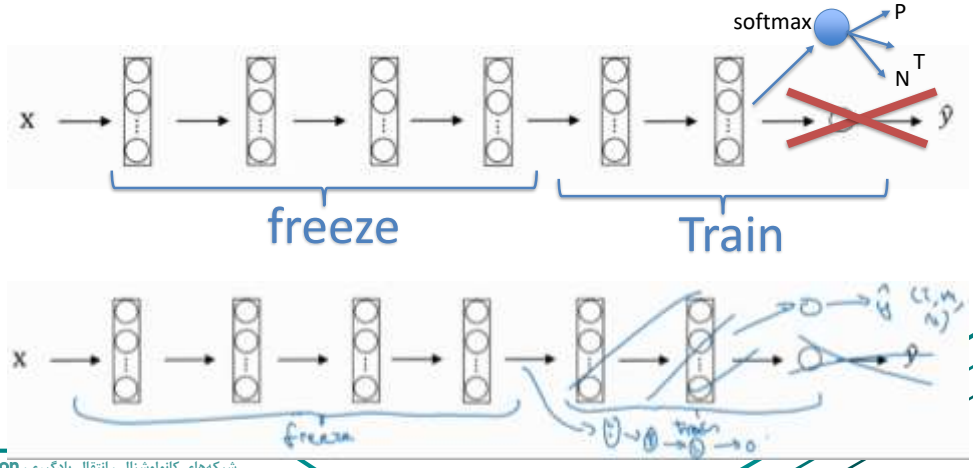
Transfer Learning


Persian Cat

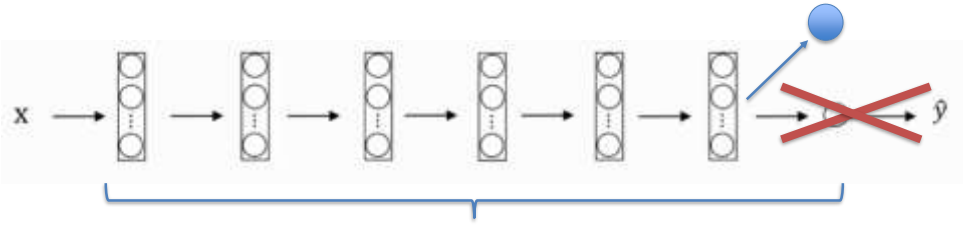

Tiger cat

?

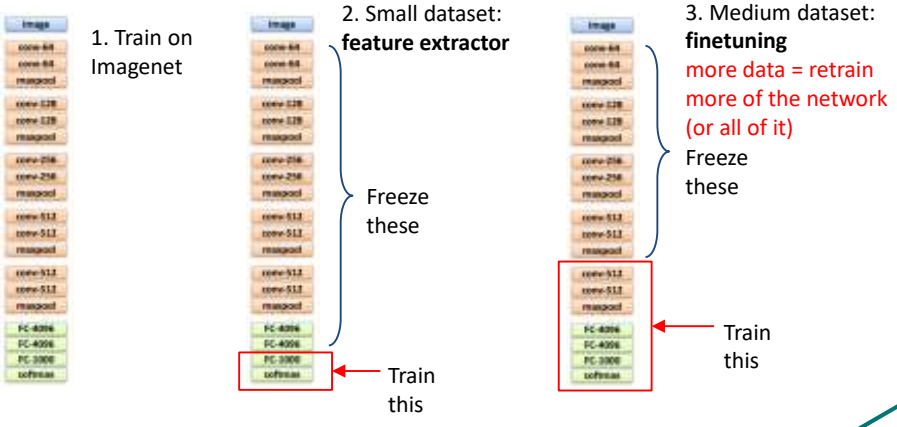
Neither



Transfer Learning



Transfer Learning with CNNs



Transfer Learning with CNNs

1. Train on Imagenet

2. Small dataset: **feature extractor**

Freeze these

Train this

3. Medium dataset: **finetuning**

more data = retrain more of the network (or all of it)

Freeze these

Train this

Transfer Learning(1)

13_Transfer_learning_part1.ipynb

Transfer Learning(2)

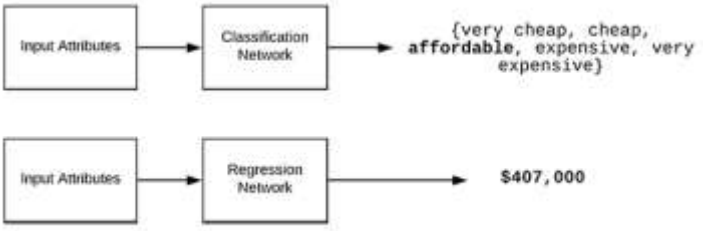
14_Transfer_learning_part2-and-Fine_tuning.ipynb

بخش ششم:

رگرسیون داده های ساختار یافته

Regression

تفاوت طبقه بندی و رگرسیون



تابع خطا یا loss برای رگرسیون ؟

Available loss functions

mean_squared_error

```
keras.losses.mean_squared_error(y_true, y_pred)
```

mean_absolute_error

```
keras.losses.mean_absolute_error(y_true, y_pred)
```

mean_absolute_percentage_error

```
keras.losses.mean_absolute_percentage_error(y_true, y_pred)
```

- https://keras.io/losses/
- https://github.com/keras-team/keras/blob/master/keras/losses.py

تابع خطا یا loss برای رگرسیون ؟

GitHub, Inc. (US) https://github.com/keras-team/keras/blob/master/

```
12
13  def mean_squared_error(y_true, y_pred):
14      return K.mean(K.square(y_pred - y_true), axis=-1)
15
16
17  def mean_absolute_error(y_true, y_pred):
18      return K.mean(K.abs(y_pred - y_true), axis=-1)
19
20
21  def mean_absolute_percentage_error(y_true, y_pred):
22      diff = K.abs((y_true - y_pred) / K.clip(K.abs(y_true),
23                                              K.epsilon(),
24                                              None))
25      return 100. * K.mean(diff, axis=-1)
```


Regression with Keras

15_ Basic-regression-with-Keras.ipynb

بخش هفتم:

مدل با چند وردی / چند خروجی

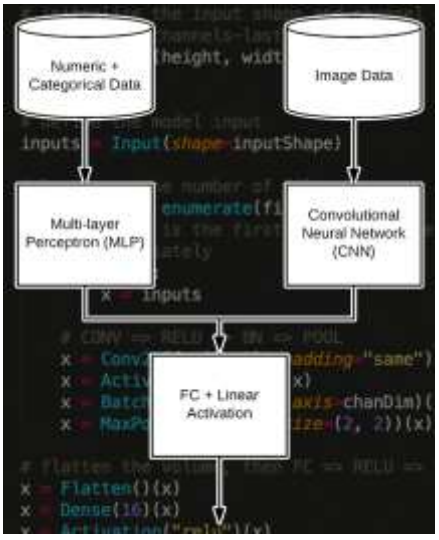
Keras Functional API

- ❑ Bedroom
- ❑ Bathroom
- ❑ Kitchen
- ❑ Frontal view of the house

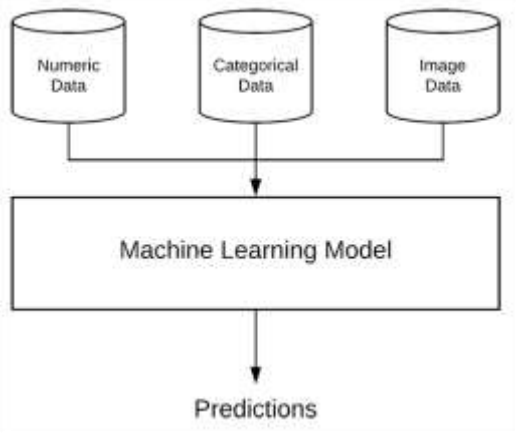
Convolutional Neural Network for House price Regression

16_cnn-and-price-regression.ipynb

یک مدل با چند ورودی



What is mixed data?



We would have multiple types of input data for a given patient, including:

1.Numeric/continuous values, such as age, heart rate, blood pressure
2.Categorical values, including gender and ethnicity
3.Image data, such as any MRI, X-ray, etc.

Keras functional API

```
model = Sequential()
model.add(Dense(8, input_shape=(10,), activation="relu"))
model.add(Dense(4, activation="relu"))
model.add(Dense(1, activation="linear"))
```

```
inputs = Input(shape=(10,))
x = Dense(8, activation="relu")(inputs)
x = Dense(4, activation="relu")(x)
x = Dense(1, activation="linear")(x)
model = Model(inputs, x)
```

imbalanced data

- **How to set class weights for imbalanced classes in Keras?**

1. **Define a dictionary with your labels and their associated weights**

```
class_weight = {0: 1.,
                1: 50.,
                2: 2.}
```

2. **Feed the dictionary as a parameter:**

model.fit(X_train, Y_train, nb_epoch=5, batch_size=32, **class_weight=class_weight**)

Our next code block handles our data imbalance issue by computing the class weights:

```
# account for skew in the labeled data
classTotals = labels.sum(axis=0)
classWeight = classTotals.max() / classTotals
```

Line 52 computes the total number of examples per class. In this case, `classTotals` will be an array: `[9475, 3690]` for "not smiling" and "smiling", respectively.

We then *scale* these totals on Line 53 to obtain the `classWeight` used to handle the class imbalance, yielding the array: `[1, 2.56]`. This weighting implies that our network will treat every instance of "smiling" as 2.56 instances of "not smiling" and helps combat the class imbalance issue by amplifying the per-instance loss by a larger weight when seeing "smiling" examples.

Now that we've computed our class weights, we can move on to partitioning our data into training and testing splits, using 80% of the data for training and 20% for testing:

```
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
(trainX, testX, trainY, testY) = train_test_split(data,
    labels, test_size=0.20, stratify=labels, random_state=42)
```

Finally, we are ready to train LeNet:

```
# initialize the model
print("[INFO] compiling model...")
model = LeNet.build(width=28, height=28, depth=1, classes=2)
model.compile(loss="binary_crossentropy", optimizer="adam",
    metrics=["accuracy"])

# train the network
print("[INFO] training network...")
H = model.fit(trainX, trainY, validation_data=(testX, testY),
    class_weight=classWeight, batch_size=64, epochs=15, verbose=1)
```

Line 62 initializes the LeNet architecture which will accept 28 × 28 single channel images.

چگونه در کراس مشکل دیتای imbalance را حل کنیم؟

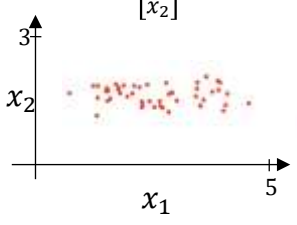
منبع :کتاب Deep Learning for Computer Vision with Python
بخش 22.2 Training the Smile CNN

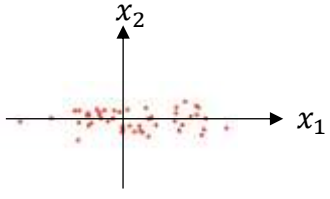
منابع

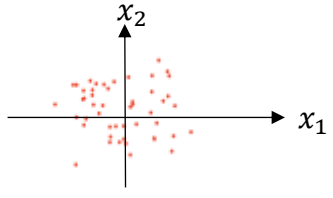
- *https://www.coursera.org/specializations/deep-learning*
- http://cs231n.stanford.edu/
- https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1
- http://blog.class.vision/winter-96-97-syllabus/
- *https://www.slideshare.net/Alirezaakhavanpour/presentation10-68048331*
- *https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/*
- *https://www.datacamp.com/courses/machine-learning-with-tree-based-models-in-python*
- *https://www.amazon.com/Deep-Learning-Python-Francois-Chollet/dp/1617294438*
- https://www.pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/

نرمال سازی training set

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$







تفریق میانگین

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$
$$x = x - \mu$$

نرمال سازی واریانس

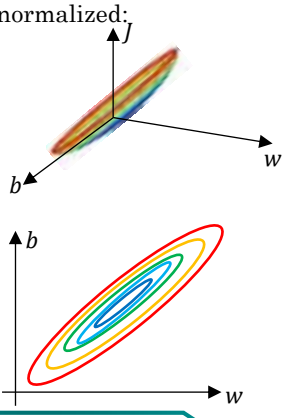
$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)})^2$$
$$x = \frac{x}{\sigma^2}$$

✓ استفاده از همین μ و σ^2 برای مجموعه داده آزمون(test)

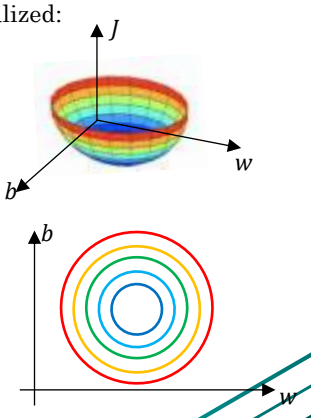
چرا ورودی را نرمال می‌کنیم؟

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

Unnormalized:



Normalized:



Batch Normalization

[Ioffe and Szegedy, 2015]

"you want unit gaussian activations? just make them so."

consider a batch of activations at some layer. To make each dimension unit gaussian, apply:

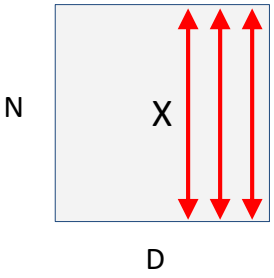
$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

this is a vanilla differentiable function...

Batch Normalization

[Ioffe and Szegedy, 2015]

"you want unit gaussian activations?
just make them so."



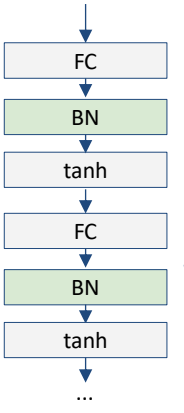
1. compute the empirical mean and variance independently for each dimension.

2. Normalize

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

[Ioffe and Szegedy, 2015]



Usually inserted after Fully Connected / (or Convolutional, as we'll see soon) layers, and before nonlinearity.

Problem: do we necessarily want a unit gaussian input to a tanh layer?

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

[Ioffe and Szegedy, 2015]

Normalize:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

And then allow the network to squash the range if it wants to:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}$$

Note, the network can learn:

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$

$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

to recover the identity mapping.

Batch Normalization

[Ioffe and Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization in a funny way, and slightly reduces the need for dropout, maybe

Batch Normalization

[Ioffe and Szegedy, 2015]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Note: at test time BatchNorm layer functions differently:

The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used.

(e.g. can be estimated during training with running averages)

Learning rate decay

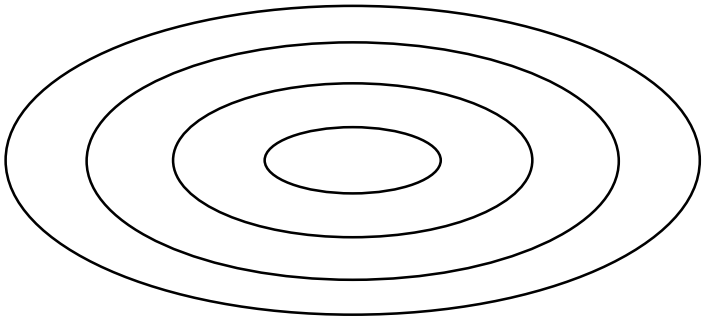
```
        self.nesterov = nesterov

    @interfaces.legacy_get_updates_support
    def get_updates(self, loss, params):
        grads = self.get_gradients(loss, params)
        self.updates = [K.update_add(self.iterations, 1)]

        lr = self.lr
        if self.initial_decay > 0:
            lr = lr * (1. / (1. + self.decay * K.cast(self.iterations,
                                                      K.dtype(self.decay))))
        # momentum
        shapes = [K.int_shape(p) for p in params]
        moments = [K.zeros(shape) for shape in shapes]
        self.weights = [self.iterations] + moments
        for p, g, m in zip(params, grads, moments):
            v = self.momentum * m - lr * g  # velocity
```

https://github.com/keras-team/keras/blob/master/keras/optimizers.py#L189

Learning rate decay



Learning rate decay