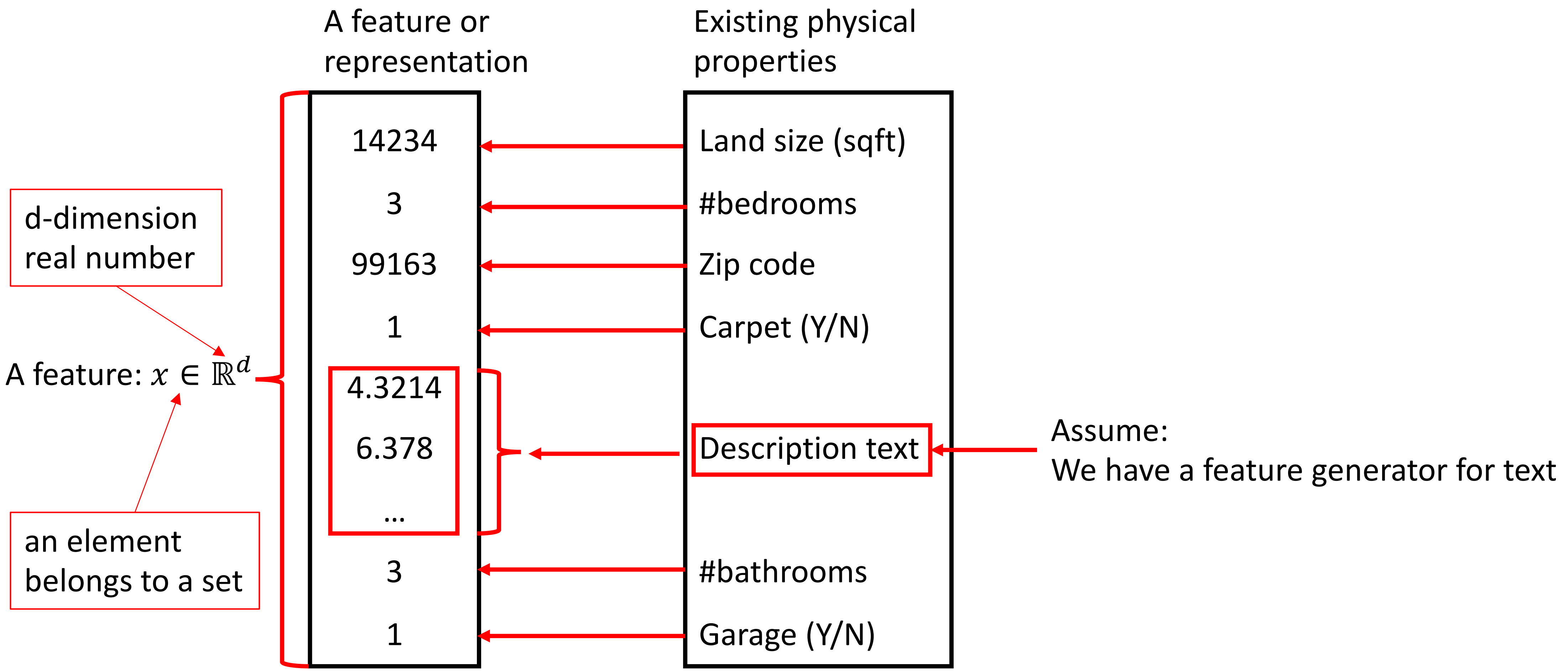
Neural Network Basics

CPT_S 434/534 Neural network design and application

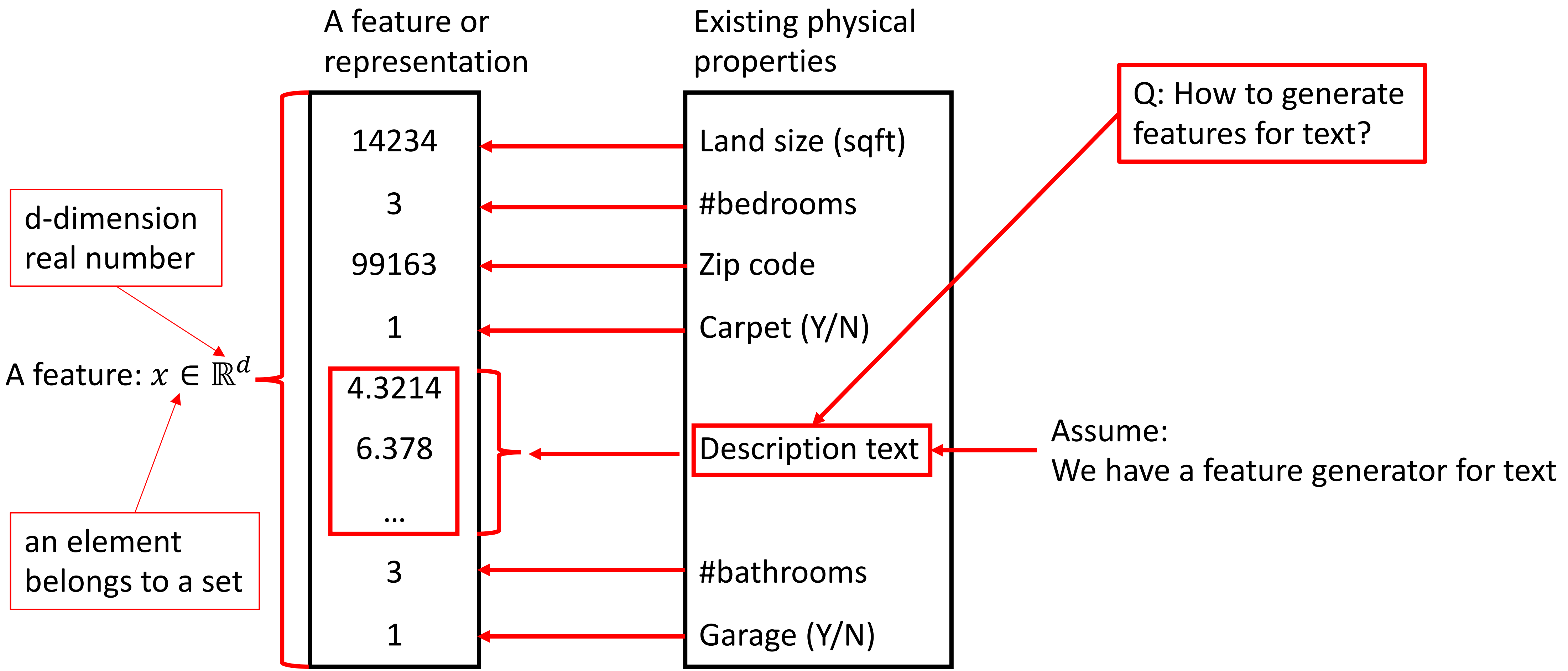
Today’s class includes

- Bag-of-words features (hand-crafted)
  - TF-IDF for text data
  - HOG for image data
- History of convolutional neural networks
  - Difference from conventional machine learning methods such as linear models (from the viewpoint of feature generation)
- Feedforward networks: a simple kind of neural networks
  - Typical structure, properties and examples

House price prediction



House price prediction



Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

total number of documents in the **corpus D** (i.e., |D|)
(A set of documents)

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

number of documents where the term t appears (i.e., $\mathrm{tf(t, d)} \neq 0$)

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

How much information the term provides (common or rare)

total number of documents in the **corpus D** (i.e., |D|)
(A set of documents)

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

number of documents where the term t appears (i.e., $\mathrm{tf(t, d)} \neq 0$)

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

How much information
the term provides
(common or rare)

total number of
documents in the
corpus D (i.e., |D|)
(A set of documents)

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

determined once
we have a corpus D

number of documents
where the term t appears
(i.e., $\mathrm{tf(t, d)} \neq 0$)

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

How much information
the term provides
(common or rare)

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

total number of
documents in the
corpus D (i.e., |D|)
(A set of documents)

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

determined once
we have a corpus D

number of documents
where the term t appears
(i.e., $\mathrm{tf}(\mathrm{t}, \mathrm{d}) \neq 0$)

tf(t, d): number of times term t
occurs in document d

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

t: a term
d: a document
D: a set of documents

How much information the term provides (common or rare)

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

total number of documents in the **corpus D** (i.e., |D|)
(A set of documents)

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

determined once we have a corpus D

number of documents where the term t appears (i.e., $\mathrm{tf(t, d)} \neq 0$)

tf(t, d): number of times term t occurs in document d (seen as a coefficient)

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

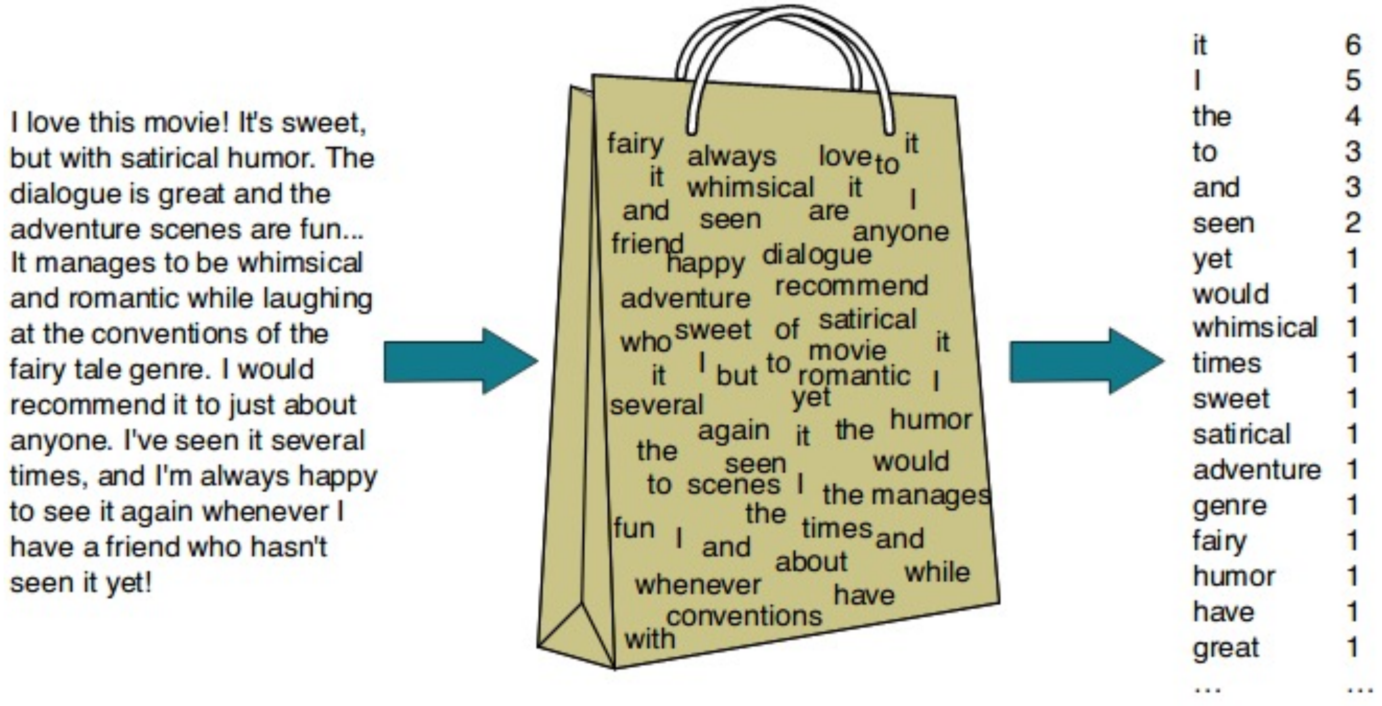


Image from https://heartbeat.fritz.ai/spam-filtering-using-bag-of-words-1c5484ff07f1

Bag-of-words features

- TF-IDF (term frequency–inverse document frequency)

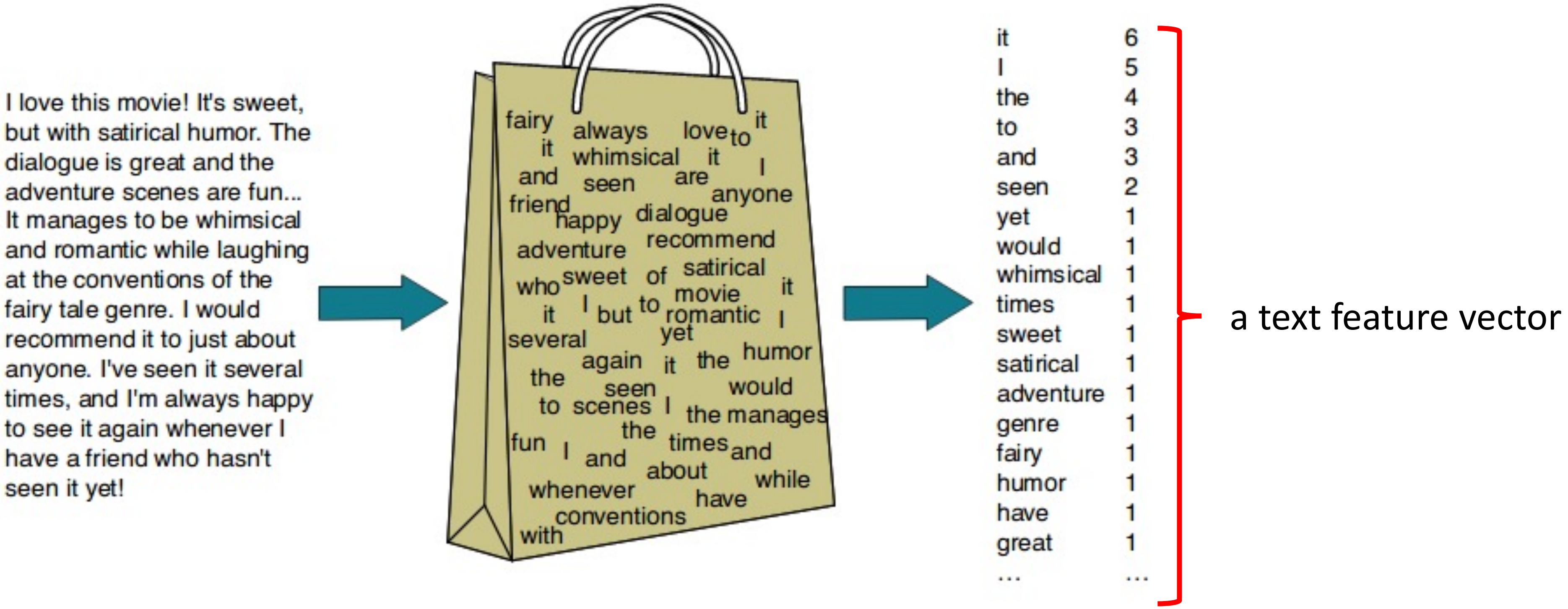


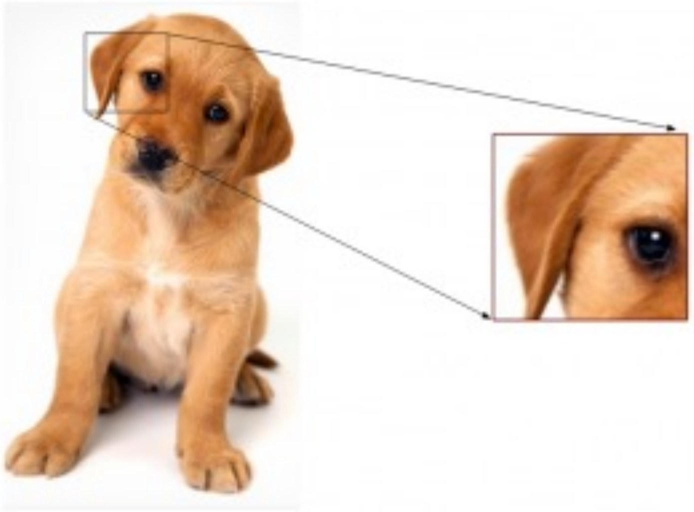
Image from https://heartbeat.fritz.ai/spam-filtering-using-bag-of-words-1c5484ff07f1

Bag-of-words features

Q: How to generate a bag-of-words feature for an image?

Histogram of oriented gradients

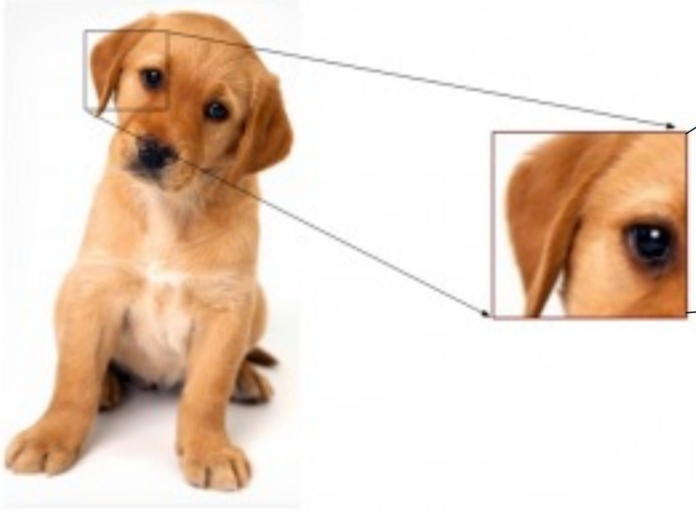
- Oriented gradients?
  - Gradients: changes in X and Y directions



Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions



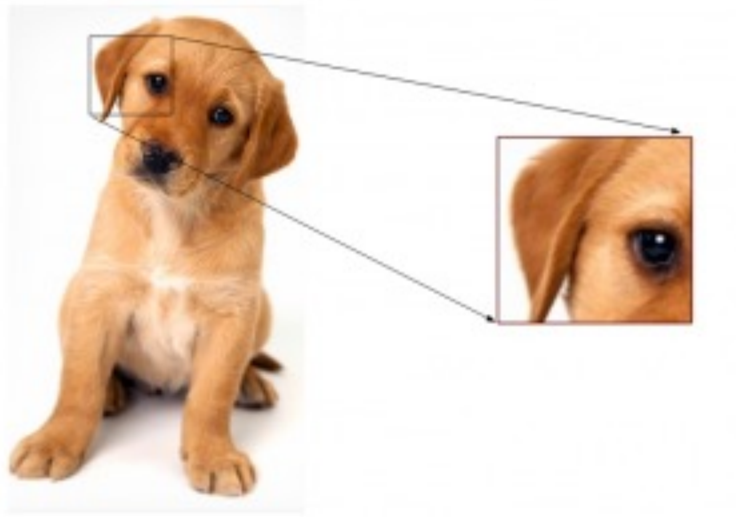
Pixel values in the image

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions



Pixel values in the image

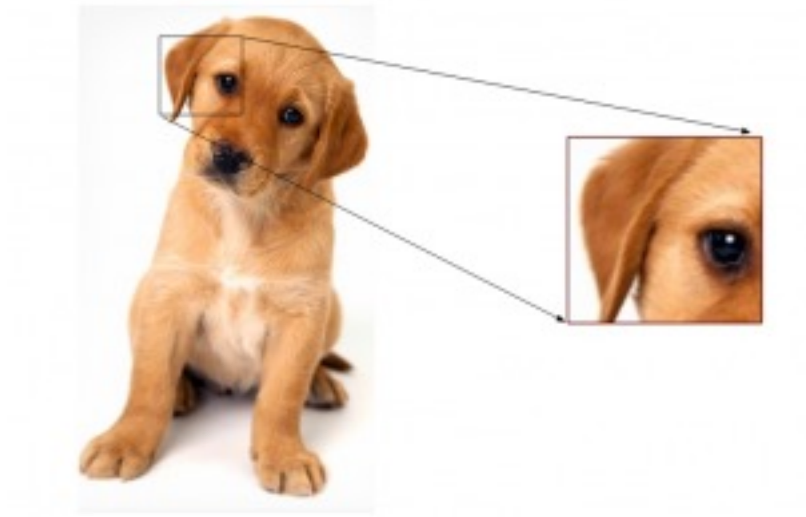
121	10	78	96	125
48	152	68	125	111
145	$\boxed{78}$	85	$\boxed{89}$	65
154	214	56	200	66
214	87	45	102	45

X direction G_x
Subtract the value on the left from the pixel value on the right:
G_x = 89-78 = 11

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions



Pixel values in the image

121	10	78	96	125
48	152	$\boxed{68}$	125	111
145	78	85	89	65
154	214	$\boxed{56}$	200	66
214	87	45	102	45

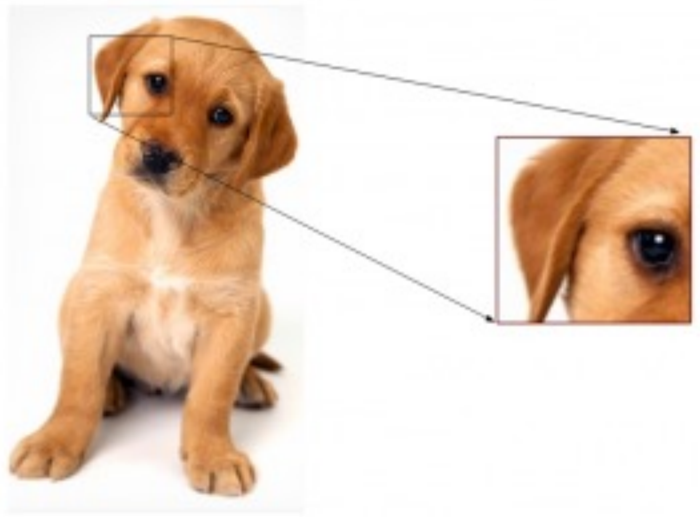
X direction G_x
Subtract the value on the left from the pixel value on the right:
G_x = 89-78 = 11

Y direction G_y
Subtract the pixel value below from the pixel value above the selected pixel:
G_y = 68-56=8

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions
  - Oriented?



Pixel values in the image

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

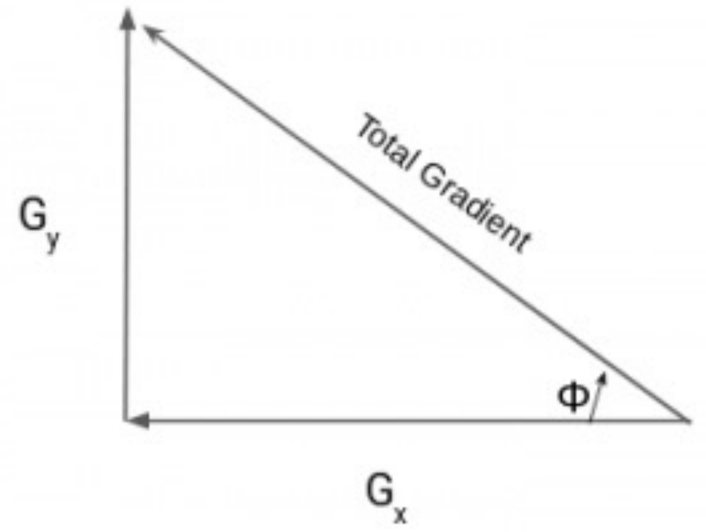
X direction G_x
Subtract the value on the left from the pixel value on the right:
G_x = 89-78 = 11

Y direction G_y
Subtract the pixel value below from the pixel value above the selected pixel:
G_y = 68-56=8

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions
  - Oriented:



Pixel values in the image

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

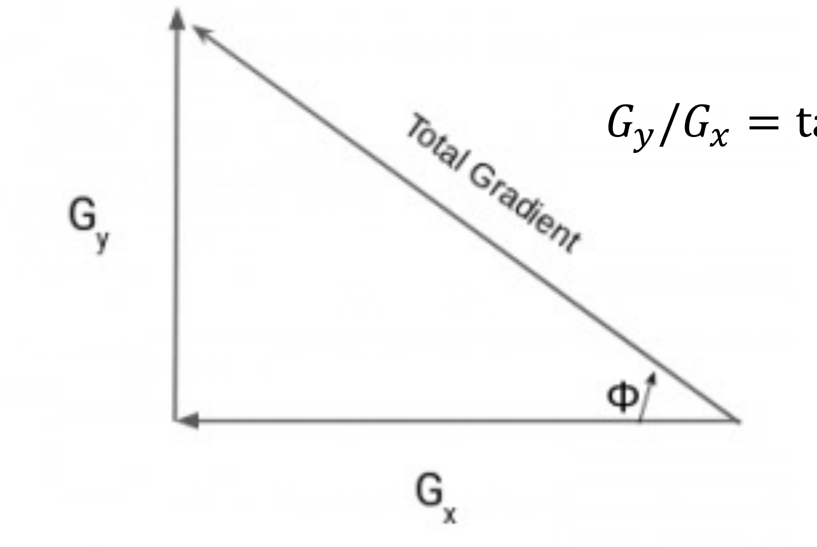
X direction G_x
Subtract the value on the left from the pixel value on the right:
G_x = 89-78 = 11

Y direction G_y
Subtract the pixel value below from the pixel value above the selected pixel:
G_y = 68-56=8

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions
  - Oriented:



$G_y/G_x = \tan(\Phi)$

Pixel values in the image

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

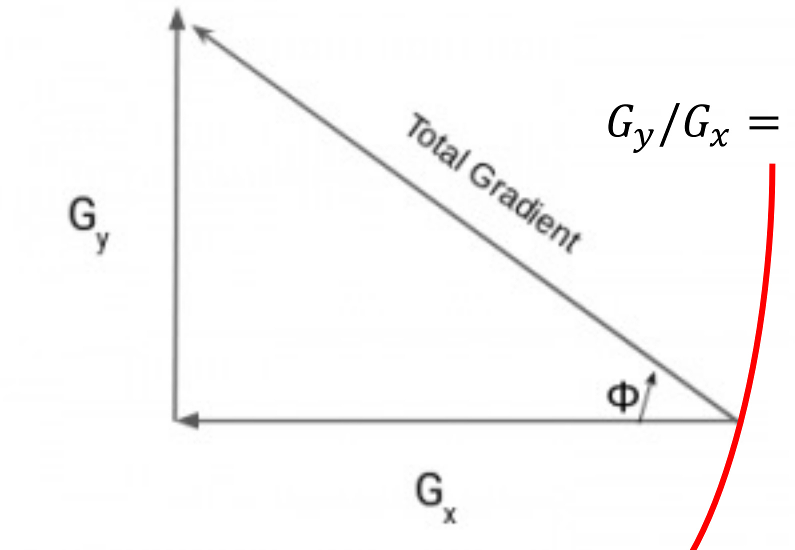
X direction G_x
Subtract the value on the left from the pixel value on the right:
G_x = 89-78 = 11

Y direction G_y
Subtract the pixel value below from the pixel value above the selected pixel:
G_y = 68-56=8

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

- Oriented gradients?
  - Gradients: changes in X and Y directions
  - Oriented:



$G_y/G_x = \tan(\Phi)$

$\Phi = \tan^{-1}(G_y/G_x)$

Pixel values in the image

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

X direction G_x
Subtract the value on the left from the pixel value on the right:
G_x = 89-78 = 11

Y direction G_y
Subtract the pixel value below from the pixel value above the selected pixel:
G_y = 68-56=8

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

121	10	78	96	125
48	152	68	125	111
145	78	(Φ=36) 85	89	65
154	214	56	200	66
214	87	45	102	45

Frequency						1										
Angle (Φ)	1	2	3	4 ...	35	$\boxed{36}$	37	38	39....	175	176	177	178	179	180

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Histogram of oriented gradients

121	10	78	96	125
48	152	68	125	111
145	78	(Φ=36) 85	89	65
154	214	56	200	66
214	87	45	102	45

How many pixels with the corresponding value of angle → a vector feature

Frequency						1									
Angle (Φ)	1	2	3	4 ...	35	$\boxed{36}$	37	38	39....	175	176	177	178	179	180

Credit for https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

Many other hand-crafted features

- Scale Invariant Feature Transform (SIFT)
- Speeded-Up Robust Feature (SURF)

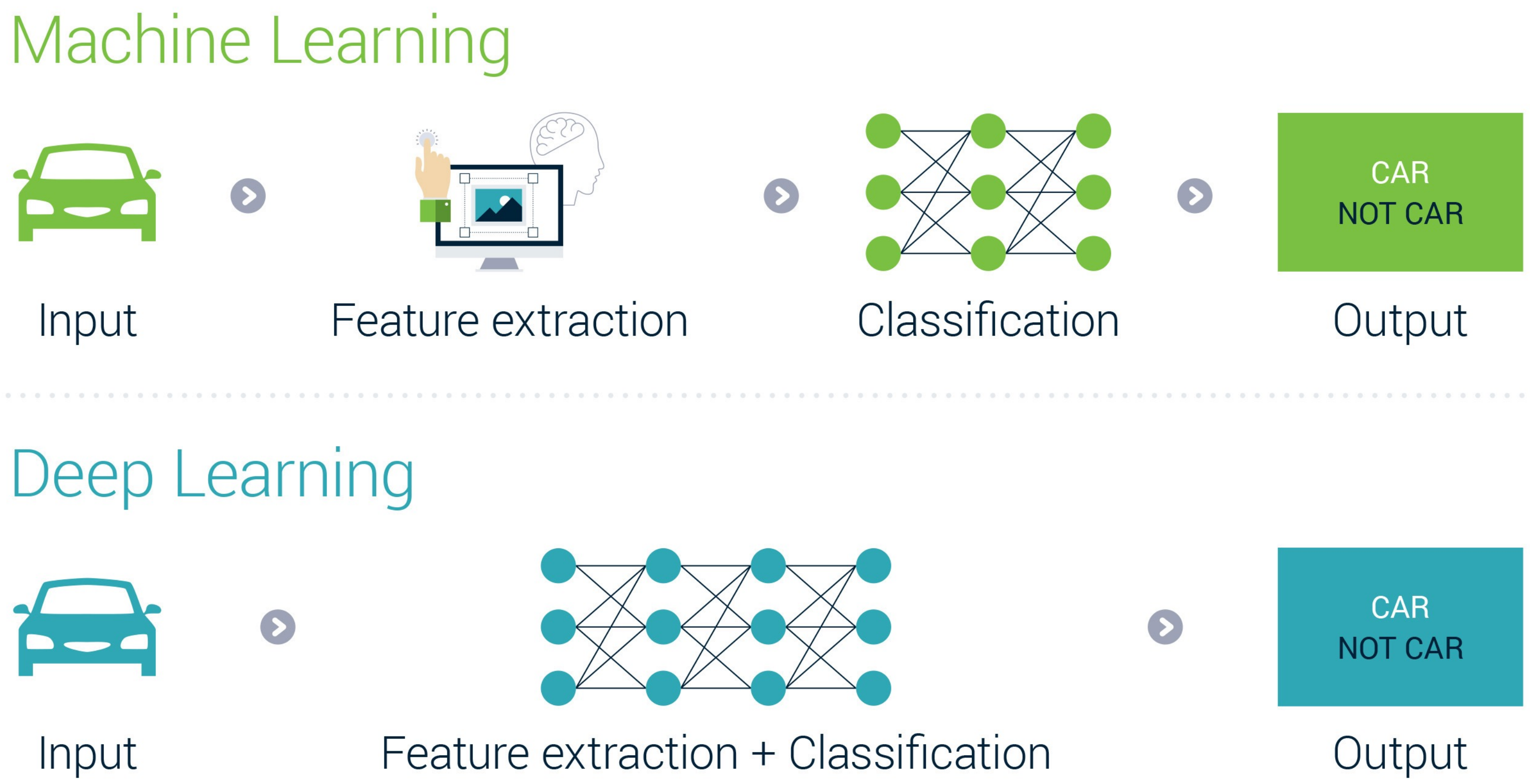
Image BOW features

- Histogram of Optical Flow (HOF)
- Motion Boundary Histogram (MBH)

Video BOW features

- Fisher vector (FV, a similarity-based function)
- Vector of Locally Aggregated Descriptors (VLAD)
- ……

End-to-end training of neural networks



LeNet-5 in 1999

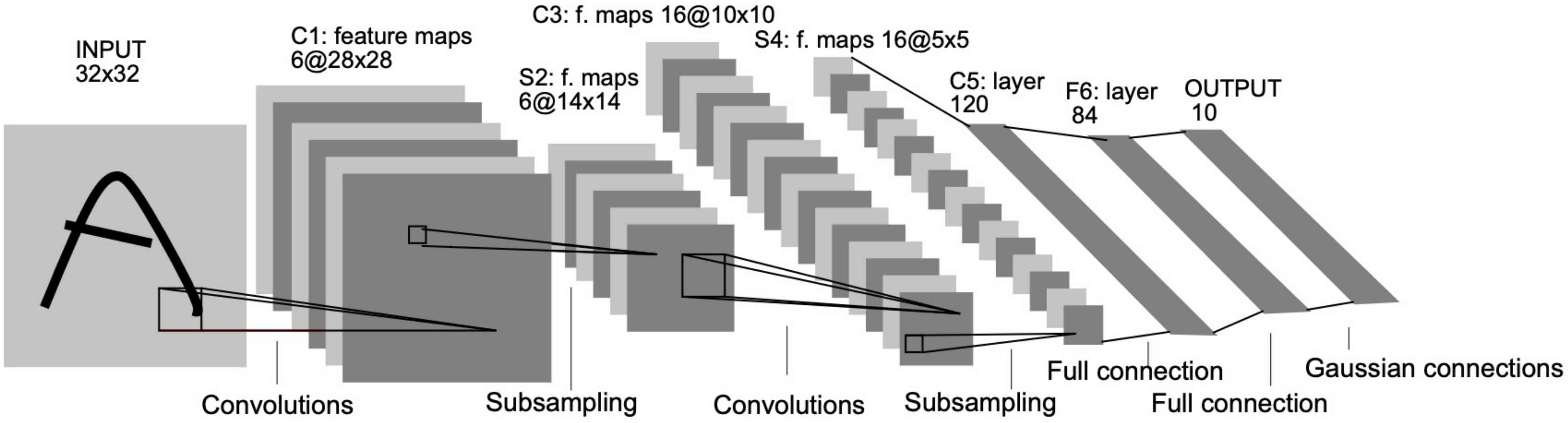


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning." In *Shape, contour and grouping in computer vision*, pp. 319-345. Springer, Berlin, Heidelberg, 1999.

ImageNet classification challenge 2012

AlexNet

Task 1

Team name	**Filename**	**Error (5 guesses)**	**Description**
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
ISI	pred_FVs_weighted.txt	0.26602	Weighted sum of scores from classifiers using each FV.
ISI	pred_FVs_summed.txt	0.26646	Naive sum of scores from classifiers using each FV.

ImageNet classification challenge 2012

AlexNet

Task 1

Team name	**Filename**	**Error (5 guesses)**	**Description**
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
ISI	pred_FVs_weighted.txt	0.26602	Weighted sum of scores from classifiers using each FV.
ISI	pred_FVs_summed.txt	0.26646	Naive sum of scores from classifiers using each FV.

AlexNet

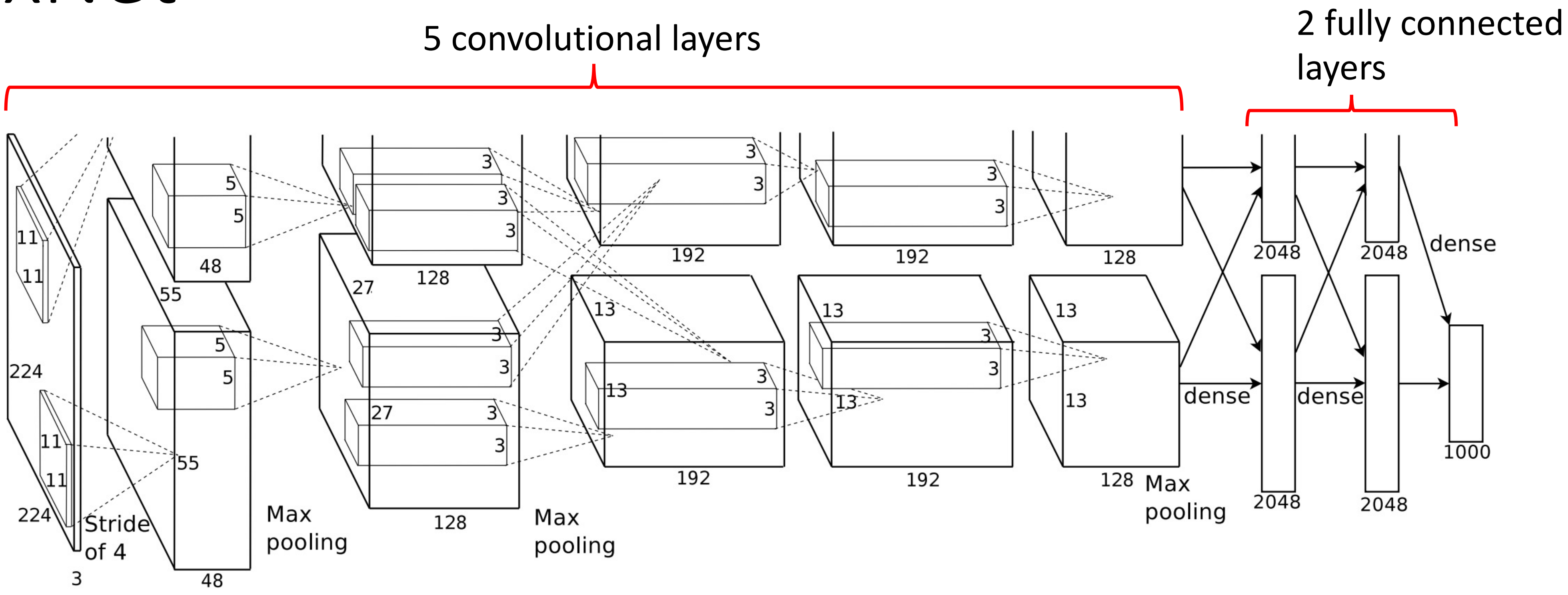
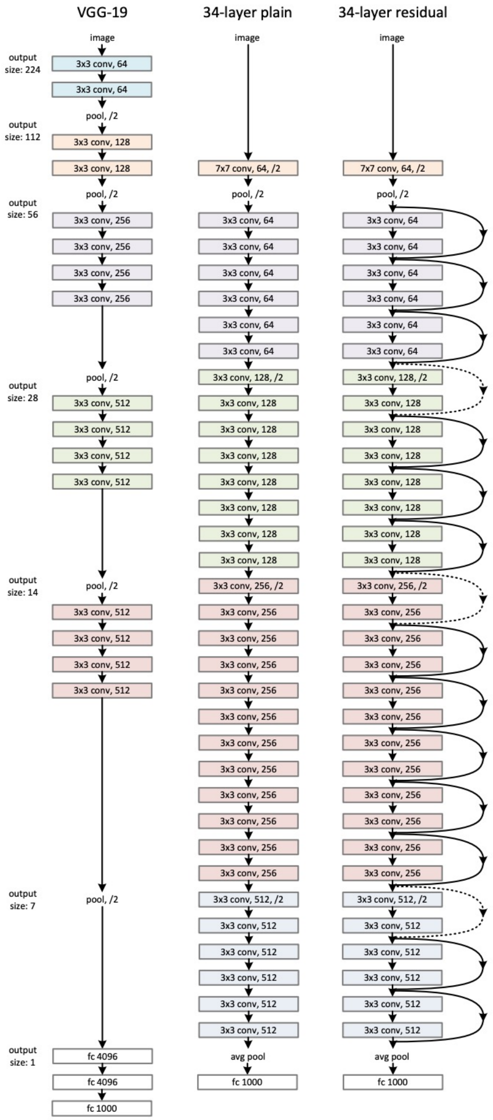


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.

VGG-19 and ResNet-34



[ResNet]

Feedforward networks

- Or multilayer perceptrons (MLPs)

$$f(w; x_i) = x_i'w + b \rightarrow y_i$$ Linear model

Feedforward networks

- Or multilayer perceptrons (MLPs)

$$f(w; x_i) = x_i{}'w + b \rightarrow y_i$$ Linear model

Weight Bias

Feedforward networks

- Or multilayer perceptrons (MLPs)

$$f(w; x_i) = x_i'w + b \rightarrow y_i$$ Linear model

Weight Bias

$$f(w; x_i)$$

Feedforward networks

- Or multilayer perceptrons (MLPs)

$$f(w; x_i) = x_i' w + b \rightarrow y_i$$ Linear model

Weight Bias

$$f(w; x_i)$$

If we composite
another function

$$f_2(f_1(w; x_i))$$

Feedforward networks

- Or multilayer perceptrons (MLPs)

$$f(w; x_i) = x_i{}'w + b \rightarrow y_i$$ Linear model

Weight Bias

$$f(w; x_i)$$

If we composite another function

$$f_2(f_1(w; x_i))$$

If we composite other functions

$$f_m\left(\underbrace{\dots\left(f_2\left(f_1(w; x_i)\right)\right)}\right) \rightarrow y_i$$

Feedforward networks

- Or multilayer perceptrons (MLPs)

$f(w; x_i) = x_i{}'w + b \to y_i$ Linear model

Weight

Bias

$f(w; x_i)$

If we composite another function

$f_2(f_1(w; x_i))$

If we composite other functions

1st layer

$f_m\left(\dots\left(f_2\left(f_1(w; x_i)\right)\right)\right) \to y_i$

final layer or output layer

Feedforward networks

- Or multilayer perceptrons (MLPs)

$$f(w; x_i) = x_i{}'w + b \to y_i$$ Linear model

Weight Bias

$$f(w; x_i)$$

If we composite another function

$$f_2(f_1(w; x_i))$$

If we composite other functions

1st layer

$$f_m\left(\underbrace{\dots\left(f_2(f_1(w; x_i))\right)}_{\text{Hidden layers}}\right) \to y_i$$

final layer or output layer

Feedforward networks

- Or multilayer perceptrons (MLPs)

Features must be extracted/generated (hand-crafted)

$$f(w; x_i) = x_i'w + b \rightarrow y_i$$

Linear model

Weight

Bias

$$f(w; x_i)$$

If we composite another function

$$f_2(f_1(w; x_i))$$

If we composite other functions

1st layer

$$f_m\underbrace{\left(\ldots\left(f_2\left(f_1(w; x_i)\right)\right)\right)}_{\text{Hidden layers}} \rightarrow y_i$$

final layer or output layer

Feedforward networks

- Or multilayer perceptrons (MLPs)

Features must be extracted/generated (hand-crafted)

$$f(w; x_i) = x_i{}'w + b \to y_i$$

Linear model

Weight

Bias

$$f(w; x_i)$$

If we composite another function

$$f_2(f_1(w; x_i))$$

If we composite other functions

1st layer

$$f_m\left(\underbrace{\ldots\left(f_2\left(f_1(w; x_i)\right)\right)}_{\text{Hidden layers}}\right) \to y_i$$

final layer or output layer

Feature extraction/generation is not necessary
Features can be learned by hidden layers

Feedforward networks

- Deep:
  - Many compositional layers

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear

e.g., activation layers

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions

$$f_m\left(\dots\left(f_2\left(f_1(w; x_i)\right)\right)\right) \to y_i$$

Composition may not preserve convexity

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex

$$f_m\left(\dots\left(f_2\left(f_1(w; x_i)\right)\right)\right) \rightarrow y_i$$

Composition may not preserve convexity

Feedforward networks

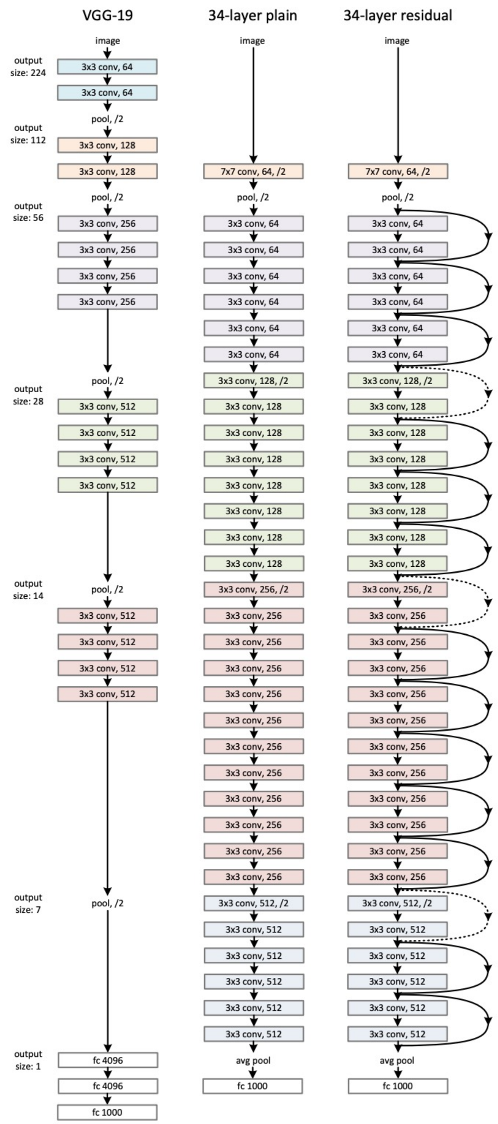
- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward **Q**: why this name?

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

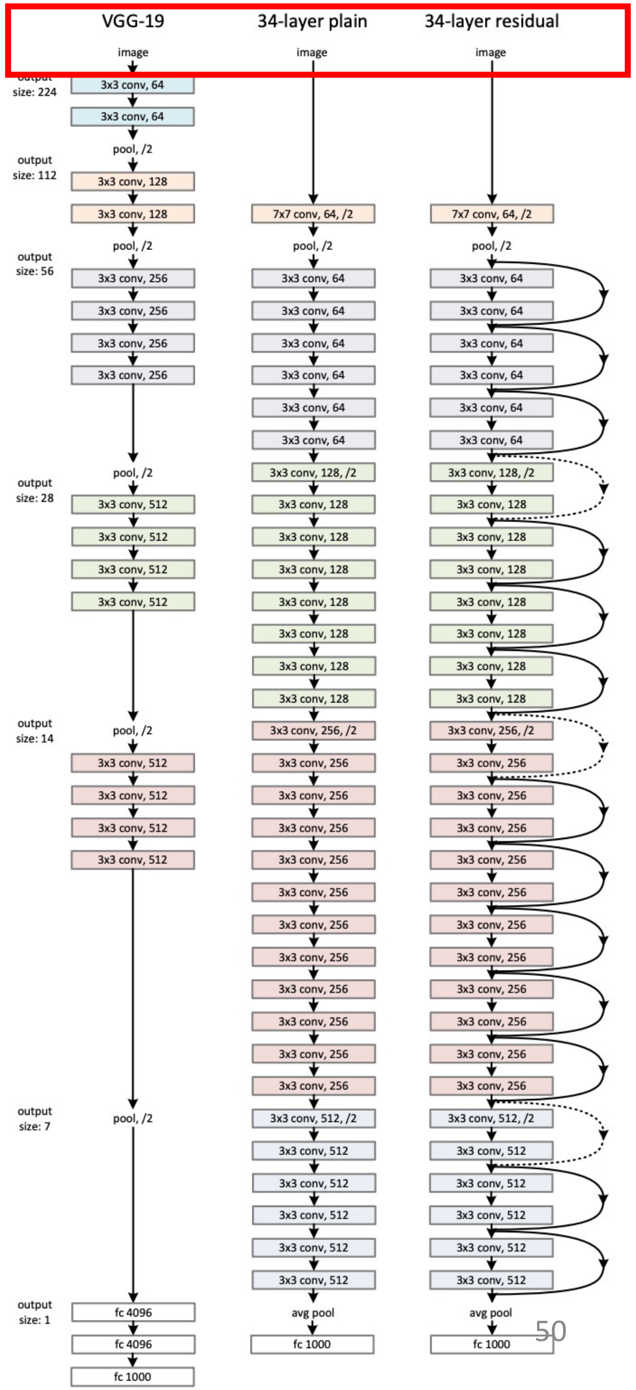


$$f_m\left(\ldots\left(f_2\left(f_1(w; x_i)\right)\right)\right) = \hat{y}_i$$

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

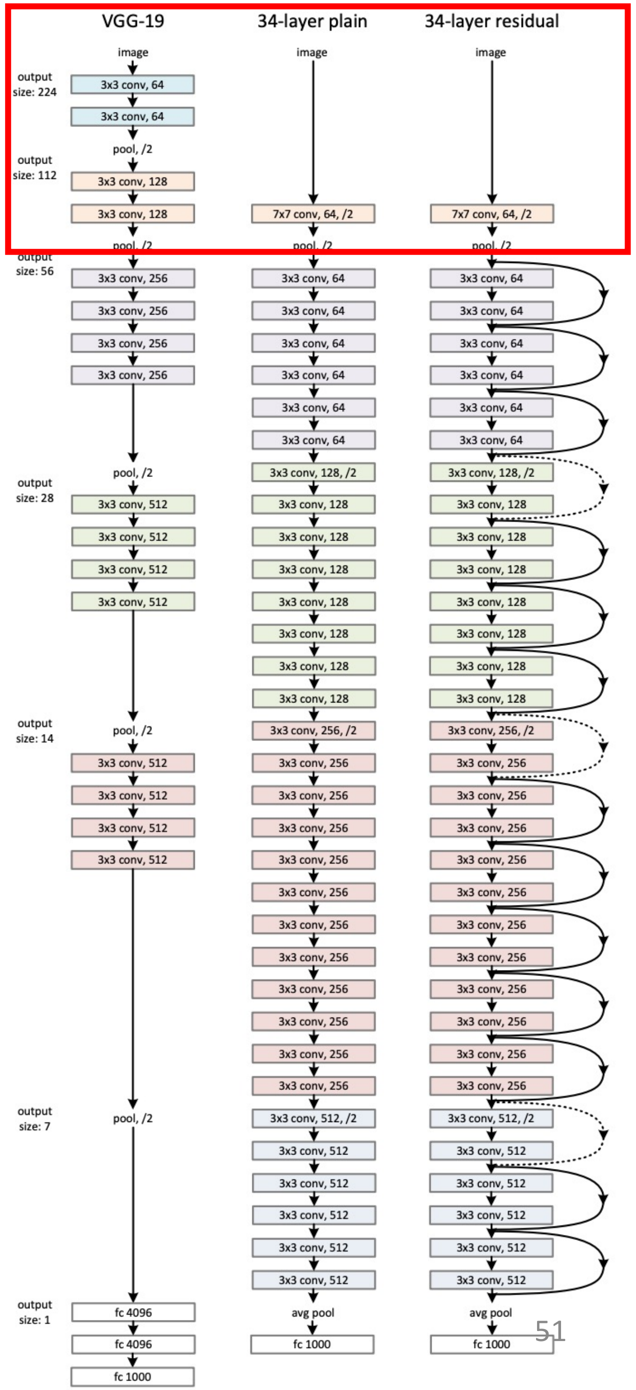
$$f_m\left(\ldots\left(f_2\left(f_1(w; {\color{red}x_i})\right)\right)\right) = \hat{y}_i$$



Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

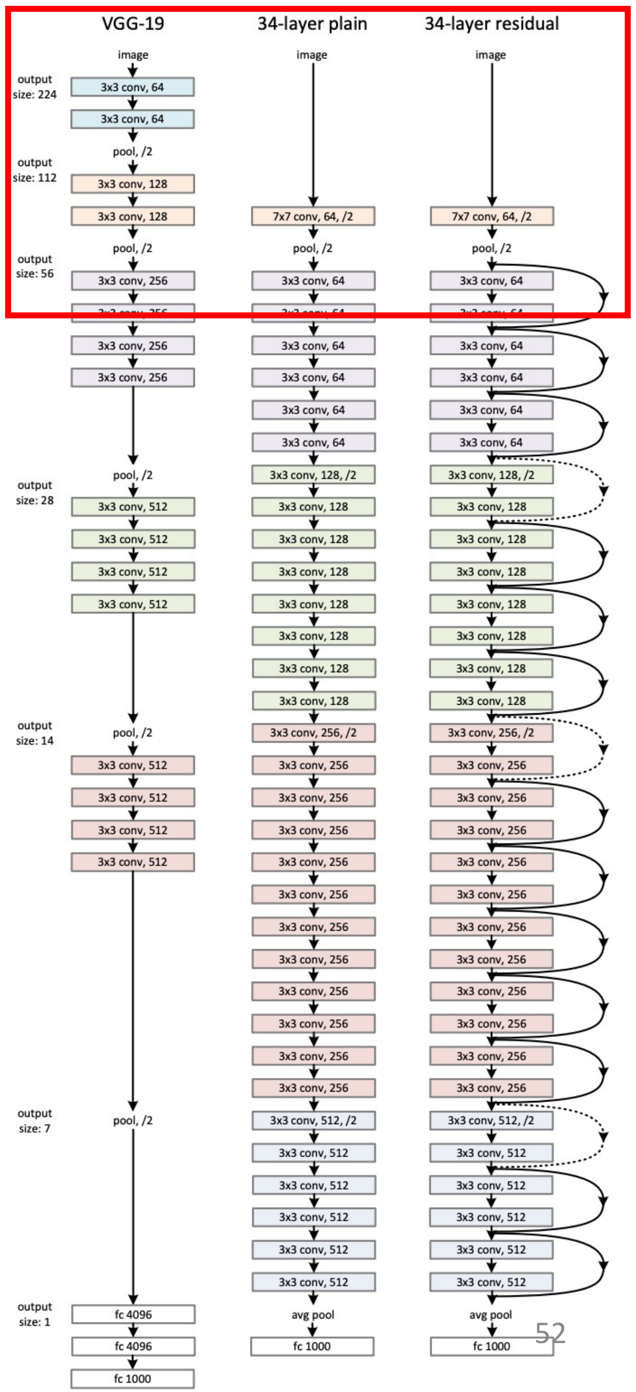
$$f_m\left(\ldots\left(f_2\left(f_1(w; x_i)\right)\right)\right) = \hat{y}_i$$



Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

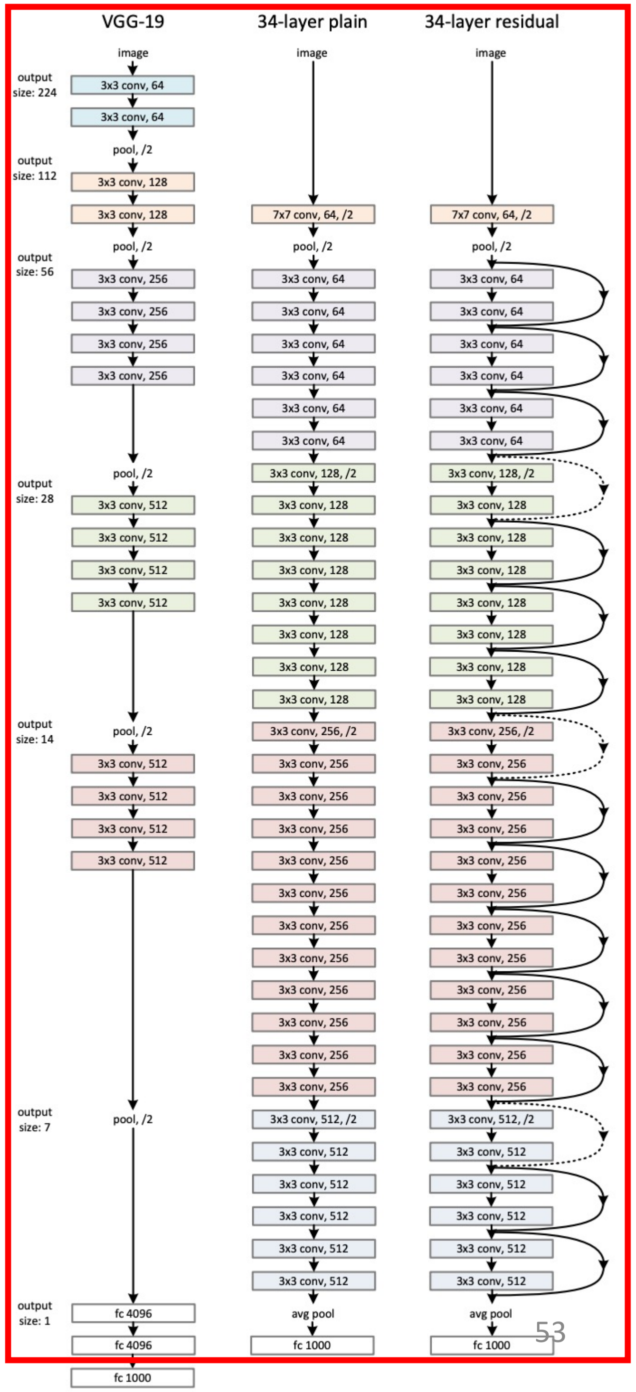
$$f_m\left(\dots\left(f_2\left(f_1(w; x_i)\right)\right)\right) = \hat{y}_i$$



Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

$$f_m\Big(\ldots\big(f_2\big(f_1(w;x_i)\big)\big)\Big) = \hat{y}_i$$



Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

Q: Any non-feedforward networks?

Feedforward networks

- Deep:
  - Many compositional layers
- Nonlinearity
  - Some functions f_i can be nonlinear
- Nonconvexity
  - Composition of functions
  - Some functions f_i can be nonconvex
- Feedforward
  - Information feedforward from input to output layer

Q: Any non-feedforward networks?
Contains circles → recurrent networks

References

- [ResNet] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.