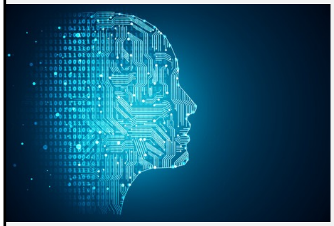
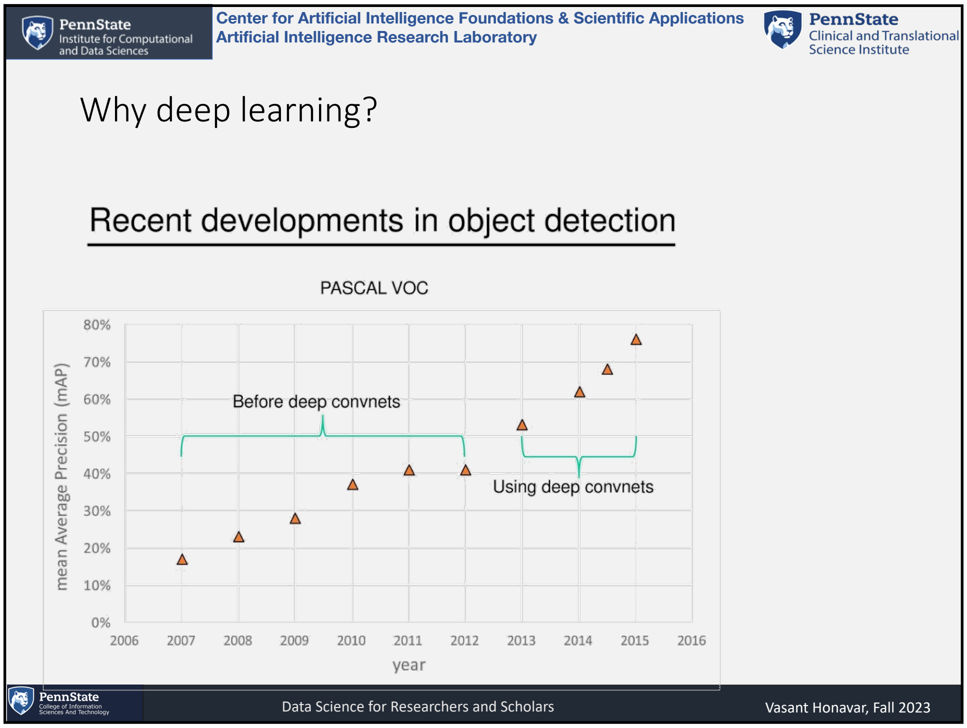


Data Science for Researchers and Scholars

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science and Engineering, Bioinformatics & Genomics, Public Health Sciences and Neuroscience
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

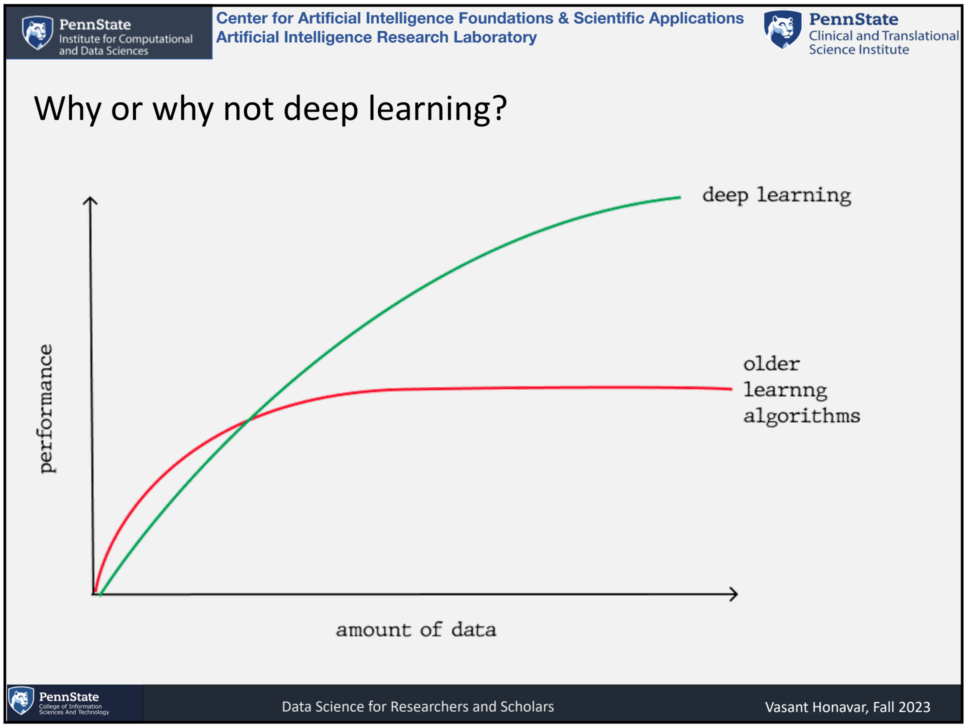
From Neural Networks to Deep Neural Networks



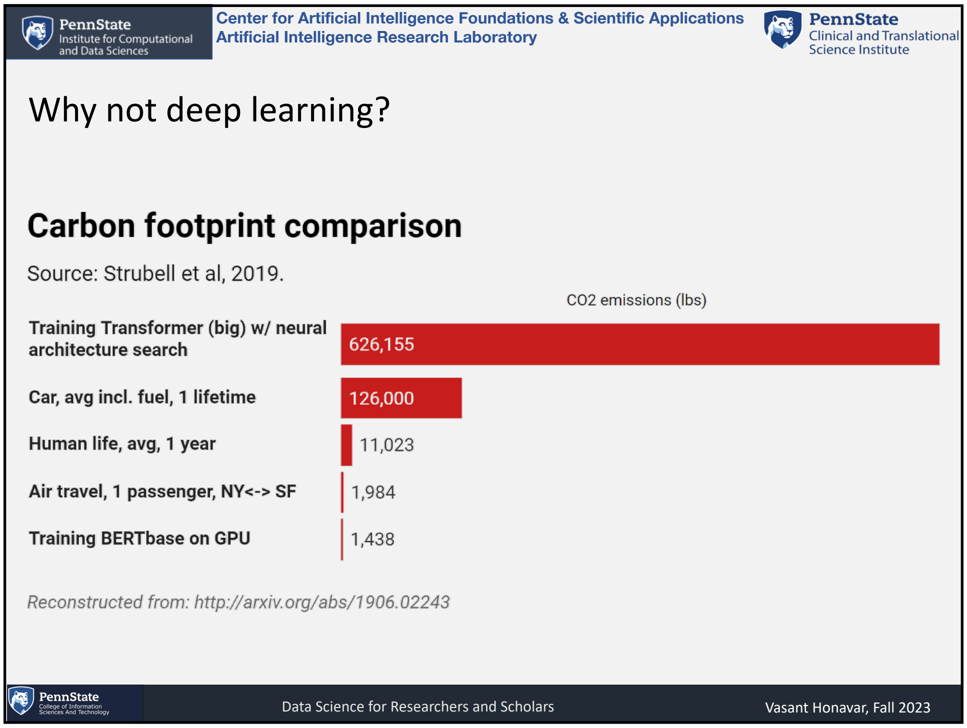
Why deep learning?

Recent developments in object detection

PASCAL VOC



Why or why not deep learning?



Why not deep learning?

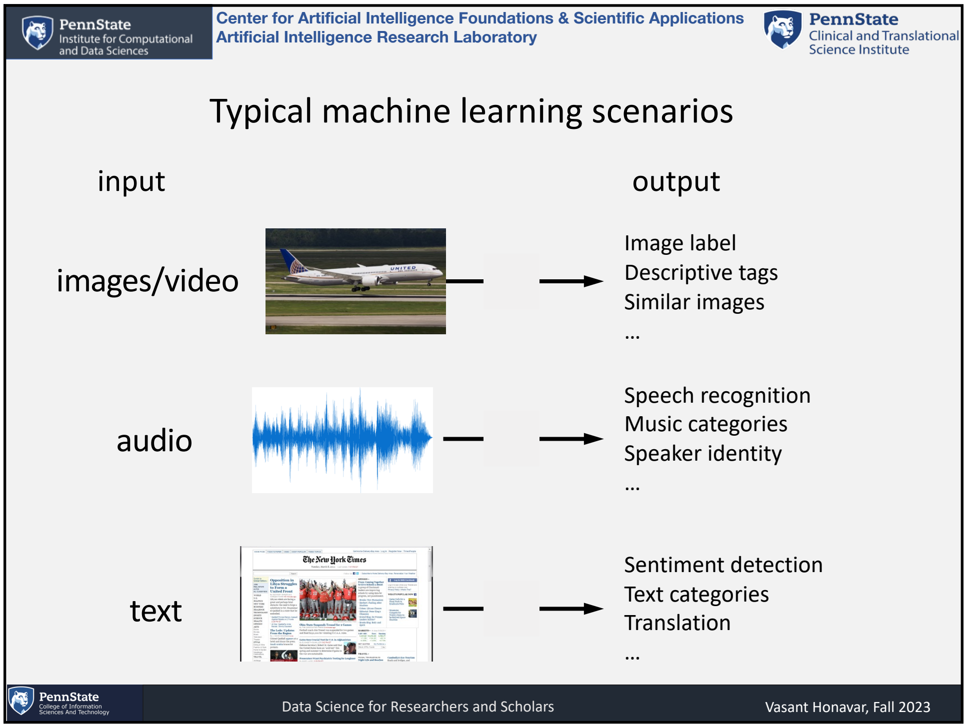
Carbon footprint comparison

Source: Strubell et al, 2019.

CO2 emissions (lbs)

Training Transformer (big) w/ neural architecture search	626,155
Car, avg incl. fuel, 1 lifetime	126,000
Human life, avg, 1 year	11,023
Air travel, 1 passenger, NY<-> SF	1,984
Training BERTbase on GPU	1,438

Reconstructed from: http://arxiv.org/abs/1906.02243



Typical machine learning scenarios

input

output

images/video

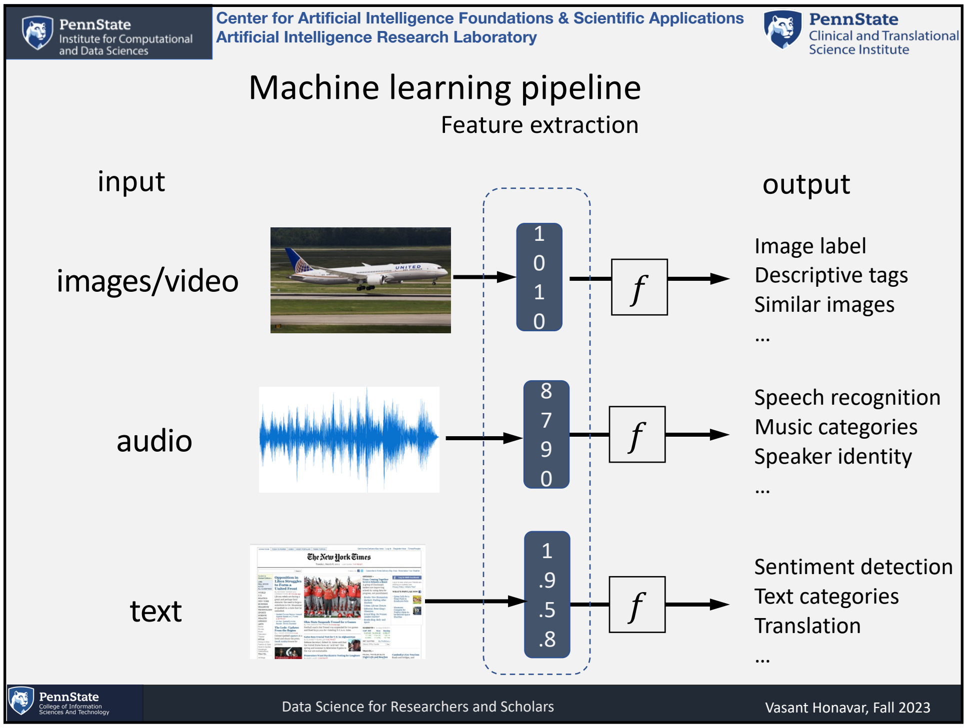
Image label
Descriptive tags
Similar images
…

audio

Speech recognition
Music categories
Speaker identity
…

text

Sentiment detection
Text categories
Translation
…



Machine learning pipeline

Feature extraction

input

output

images/video

1
0
1
0

f

Image label
Descriptive tags
Similar images
…

audio

8
7
9
0

f

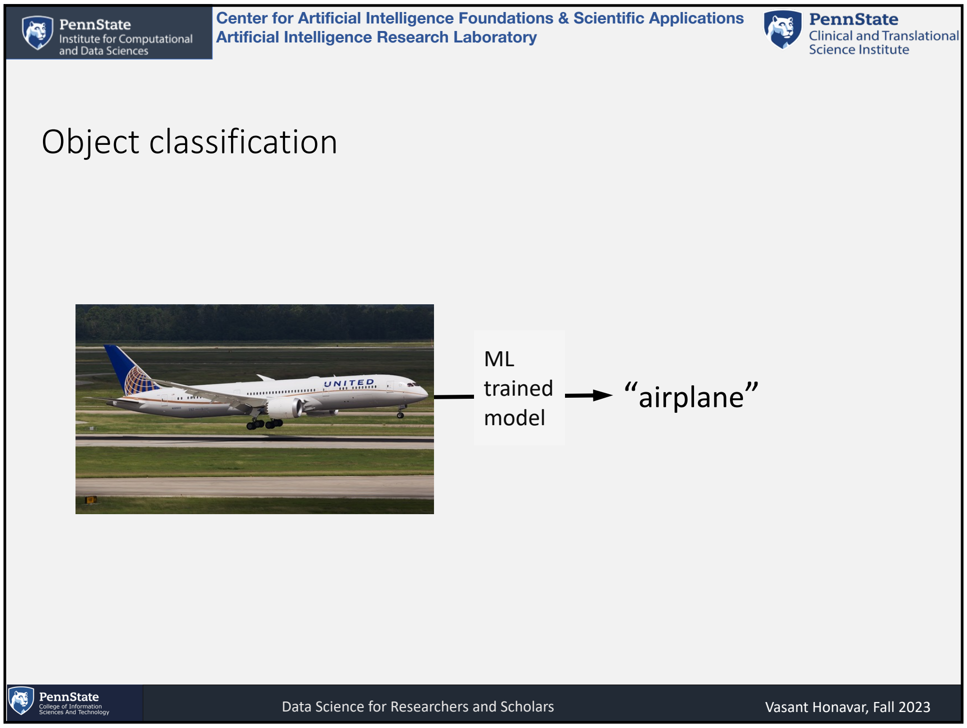
Speech recognition
Music categories
Speaker identity
…

text

1
.9
.5
.8

f

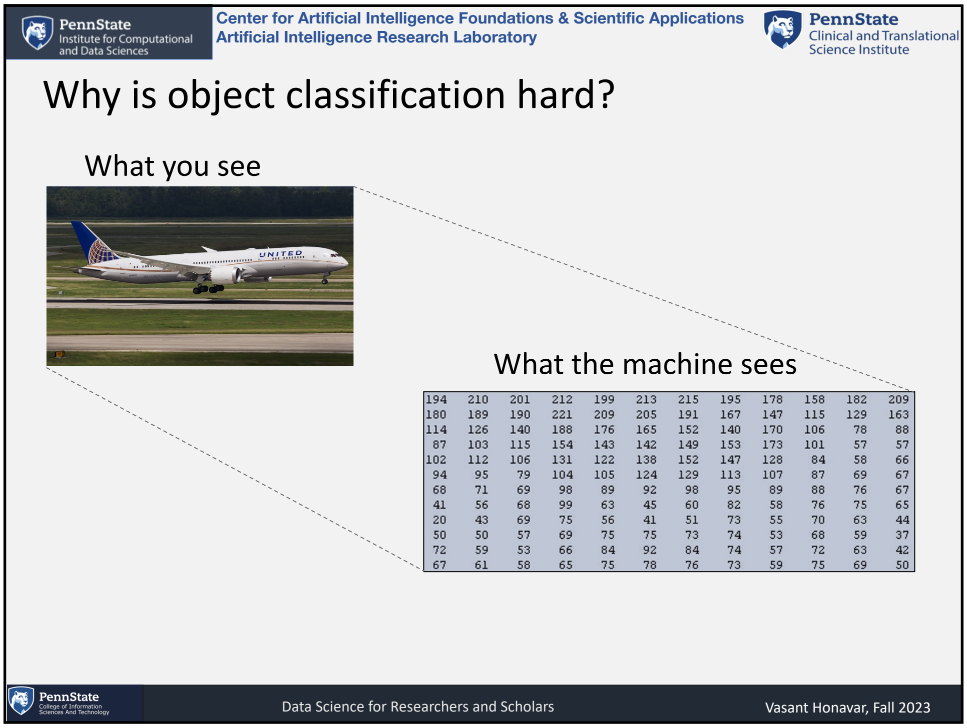
Sentiment detection
Text categories
Translation
…



Object classification

ML
trained
model

"airplane"

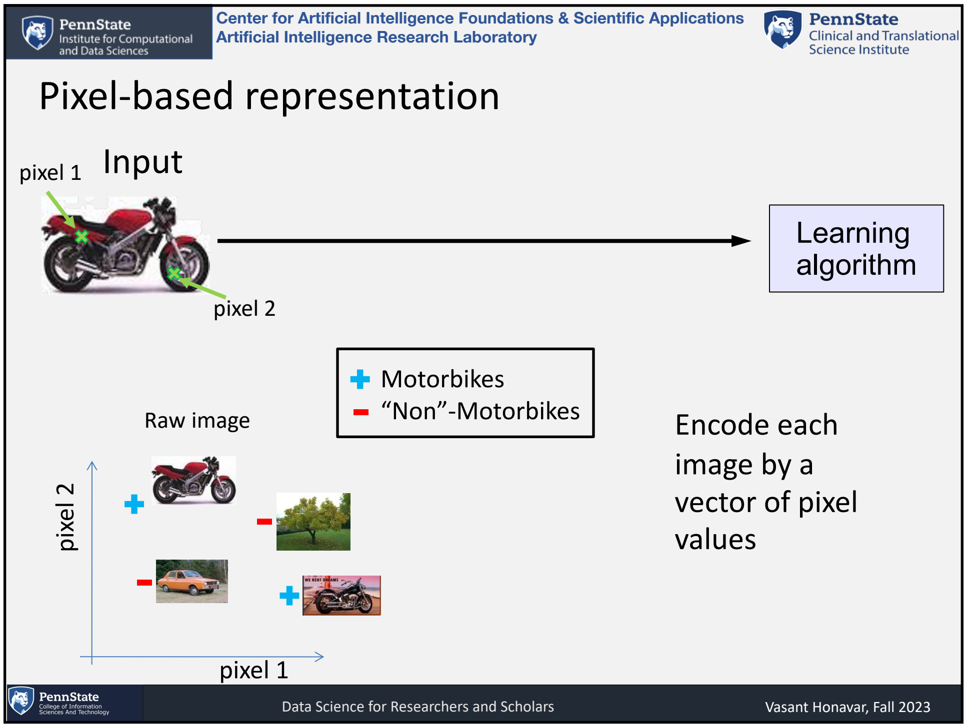


Why is object classification hard?

What you see

What the machine sees

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50



Pixel-based representation

pixel 1

Input

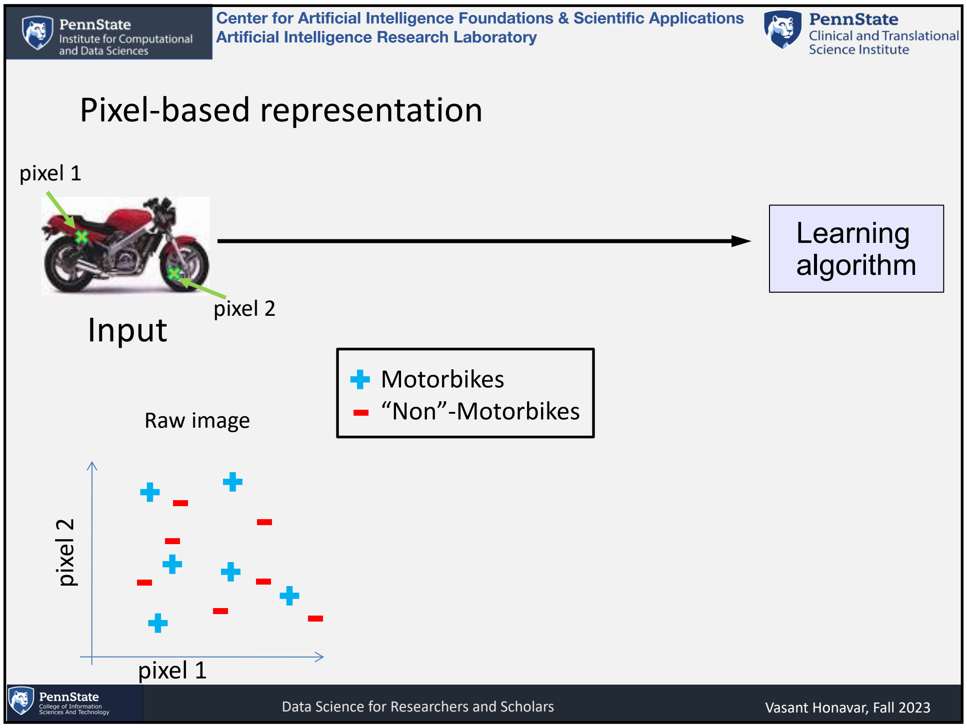
pixel 2

Learning algorithm

Raw image

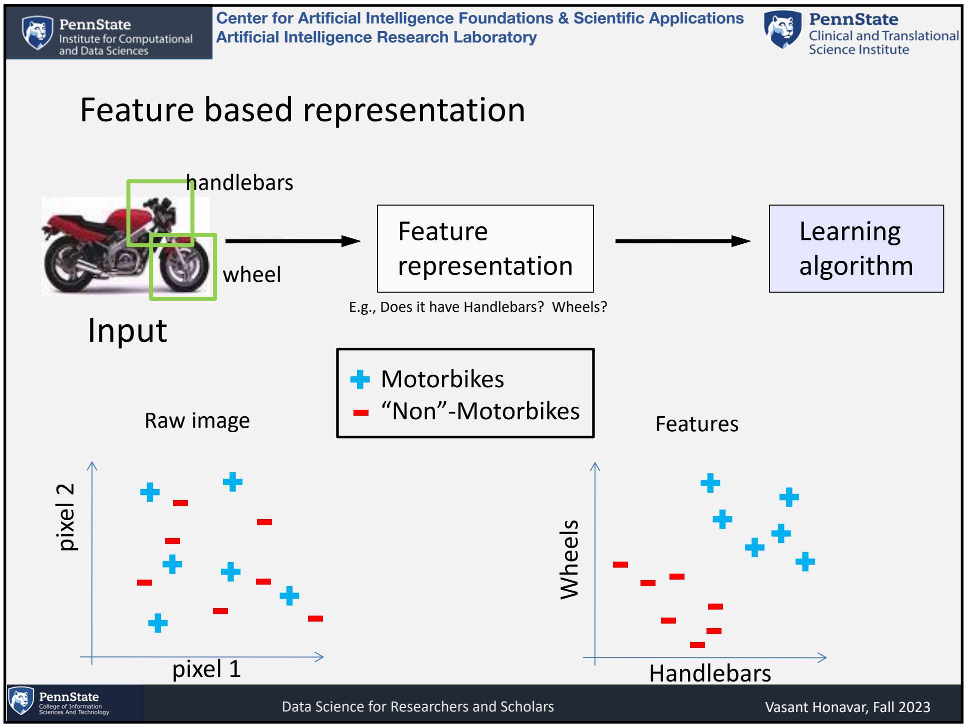
+ Motorbikes
− "Non"-Motorbikes

Encode each image by a vector of pixel values



Pixel-based representation

- ✚ Motorbikes
- ▬ “Non”-Motorbikes



Feature based representation

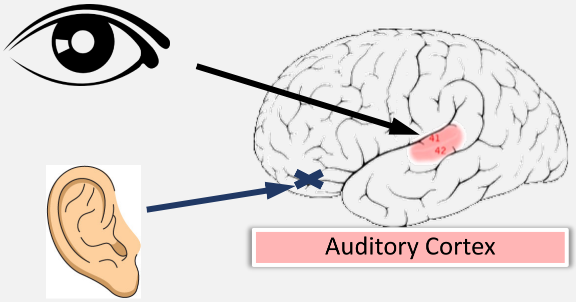
E.g., Does it have Handlebars? Wheels?

- ✚ Motorbikes
- ▬ “Non”-Motorbikes

Traditional Machine learning

- Machine learning 30 years ago relied on feature engineering
- Feature engineering is hard!
- It would be nice if we can avoid ad hoc feature engineering
- Kernel machines offer one solution
- Deep learning offers another

The brain: inspiration for deep learning

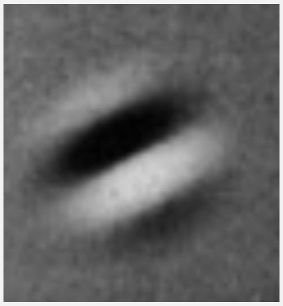


Auditory cortex learns to see!

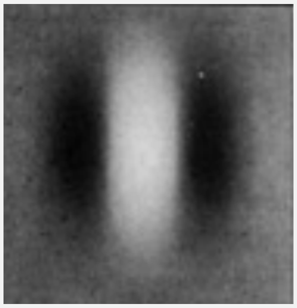
Sur, M., Garraghty, P.E. and Roe, A.W., 1988. Experimentally induced visual projections into auditory thalamus and cortex. Science, 242(4884), pp.1437-41.

First stage of visual processing: V1

- V1 is the first stage of visual processing in the brain
- Neurons in V1 act as edge detectors



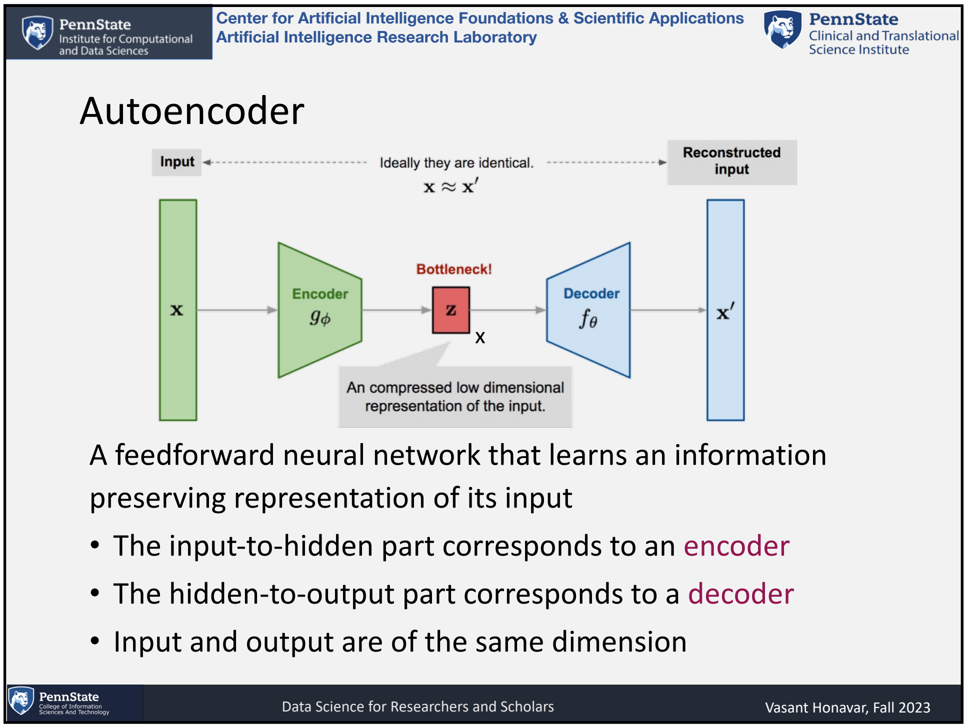
Neuron #1 of visual cortex (model)



Neuron #2 of visual cortex (model)

Basic idea of deep learning

- Involves representation learning or unsupervised feature learning (with subtle distinctions)
- Learn features from data even without knowing the task to be performed?
- Then, stack such representation learning layers to obtain ‘deep’ networks

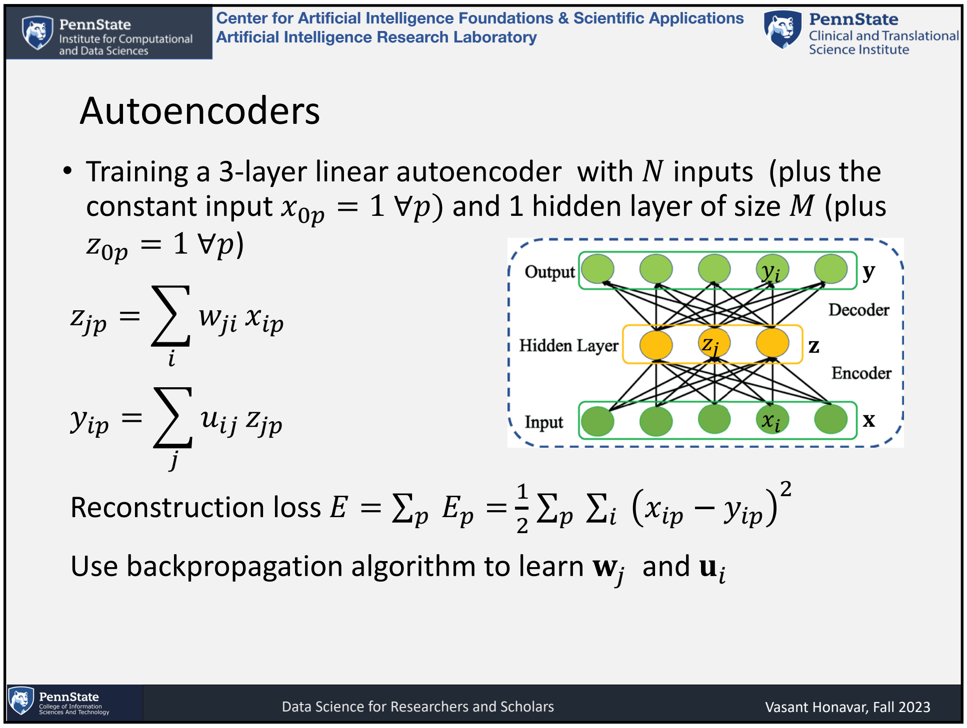


Autoencoder

A feedforward neural network that learns an information preserving representation of its input

- The input-to-hidden part corresponds to an encoder
- The hidden-to-output part corresponds to a decoder
- Input and output are of the same dimension

17



Autoencoders

- Training a 3-layer linear autoencoder with N inputs (plus the constant input $x_{0p} = 1 \ \forall p$) and 1 hidden layer of size M (plus $z_{0p} = 1 \ \forall p$)

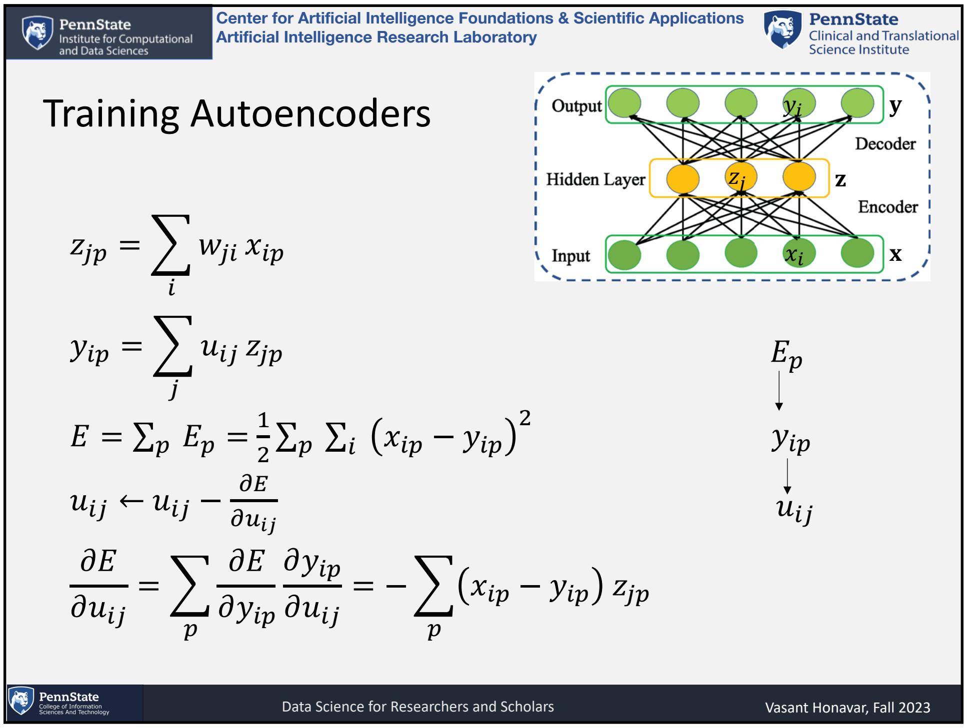
$$z_{jp} = \sum_i w_{ji} \, x_{ip}$$

$$y_{ip} = \sum_j u_{ij} \, z_{jp}$$

Reconstruction loss $E = \sum_p E_p = \frac{1}{2} \sum_p \sum_i \left(x_{ip} - y_{ip}\right)^2$

Use backpropagation algorithm to learn $\mathbf{w}_j$ and $\mathbf{u}_i$

18



Training Autoencoders

$$z_{jp} = \sum_i w_{ji}\, x_{ip}$$

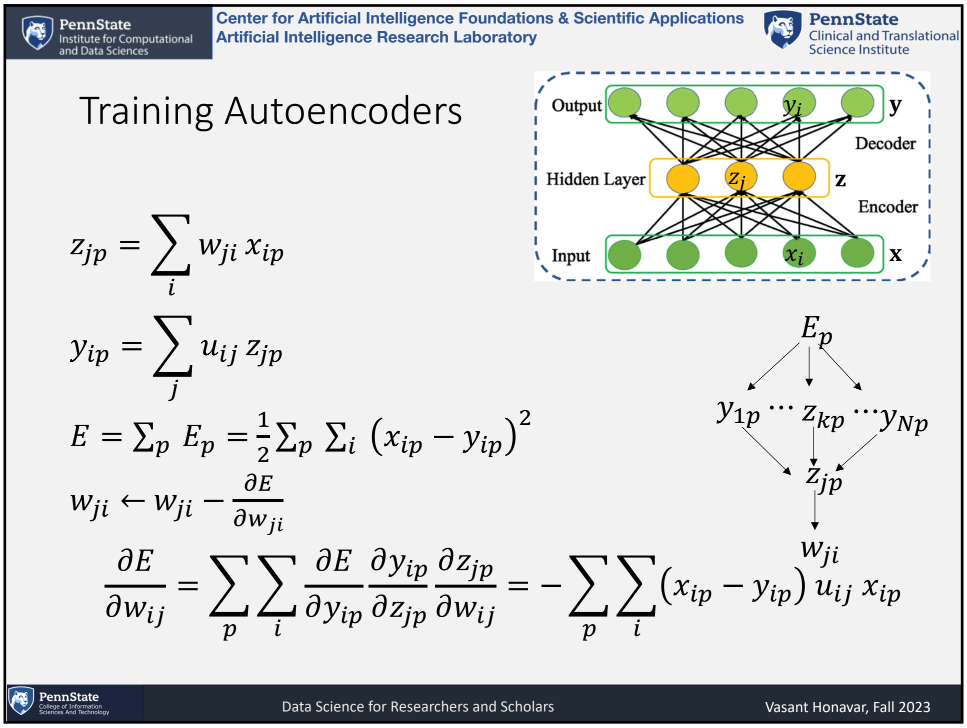
$$y_{ip} = \sum_j u_{ij}\, z_{jp}$$

$$E = \sum_p E_p = \tfrac{1}{2}\sum_p \sum_i \left(x_{ip} - y_{ip}\right)^2$$

$$u_{ij} \leftarrow u_{ij} - \frac{\partial E}{\partial u_{ij}}$$

$$\frac{\partial E}{\partial u_{ij}} = \sum_p \frac{\partial E}{\partial y_{ip}} \frac{\partial y_{ip}}{\partial u_{ij}} = -\sum_p \left(x_{ip} - y_{ip}\right) z_{jp}$$

$$\begin{tikzcd} E_p \arrow[d] \\ y_{ip} \arrow[d] \\ u_{ij} \end{tikzcd}$$



Training Autoencoders

$$z_{jp} = \sum_i w_{ji}\, x_{ip}$$

$$y_{ip} = \sum_j u_{ij}\, z_{jp}$$

$$E = \sum_p E_p = \tfrac{1}{2}\sum_p \sum_i \left(x_{ip} - y_{ip}\right)^2$$

$$w_{ji} \leftarrow w_{ji} - \frac{\partial E}{\partial w_{ji}}$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_p \sum_i \frac{\partial E}{\partial y_{ip}} \frac{\partial y_{ip}}{\partial z_{jp}} \frac{\partial z_{jp}}{\partial w_{ij}} = -\sum_p \sum_i \left(x_{ip} - y_{ip}\right) u_{ij}\, x_{ip}$$

$$\begin{tikzcd} & E_p \arrow[dl] \arrow[d] \arrow[dr] & \\ y_{1p} \cdots \arrow[dr] & z_{kp} \arrow[d] & \cdots y_{Np} \arrow[dl] \\ & z_{jp} \arrow[d] & \\ & w_{ji} & \end{tikzcd}$$

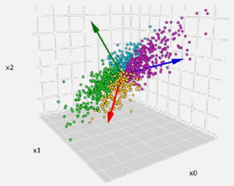
Autoencoders

- What is the point of learning a mapping that reproduces the input?
  - If the hidden layer has lower dimension than the input, the network is forced to learn a low-dimensional information preserving representation of the input
  - Once such a representation is learned, we can discard the decoder and use the encoder to extract features from the input data that can then be used to train a classification or regression model

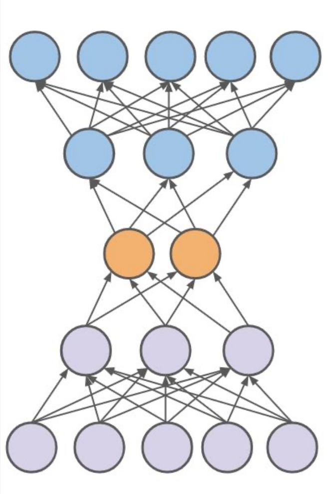
---

Autoencoders

- The autoencoder considered above is a linear autoencoder because it uses a linear function in the hidden layer
- A linear autoencoder learns an encoding that mimics principal component analysis
  - the number of hidden nodes correspond to the number of principal components
- We can obtain a non-linear autoencoder by replacing the linear function with a sigmoid function in the hidden layer
- We can use multiple hidden layers to learn complex autoencoder mapping



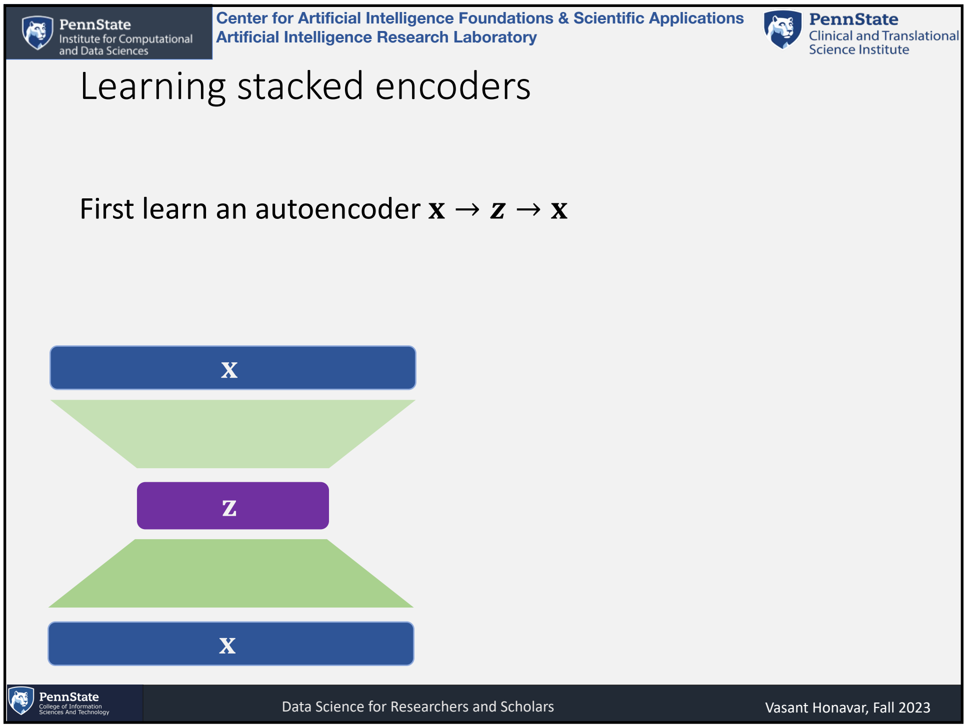
Stacked autoencoders



- We can learn hierarchical representations of input data using multi-layer nonlinear encoder
- But as we increase the number of layers, training becomes slow
- Solution: Learn a multi-layer encoder one layer at a time
  - How?

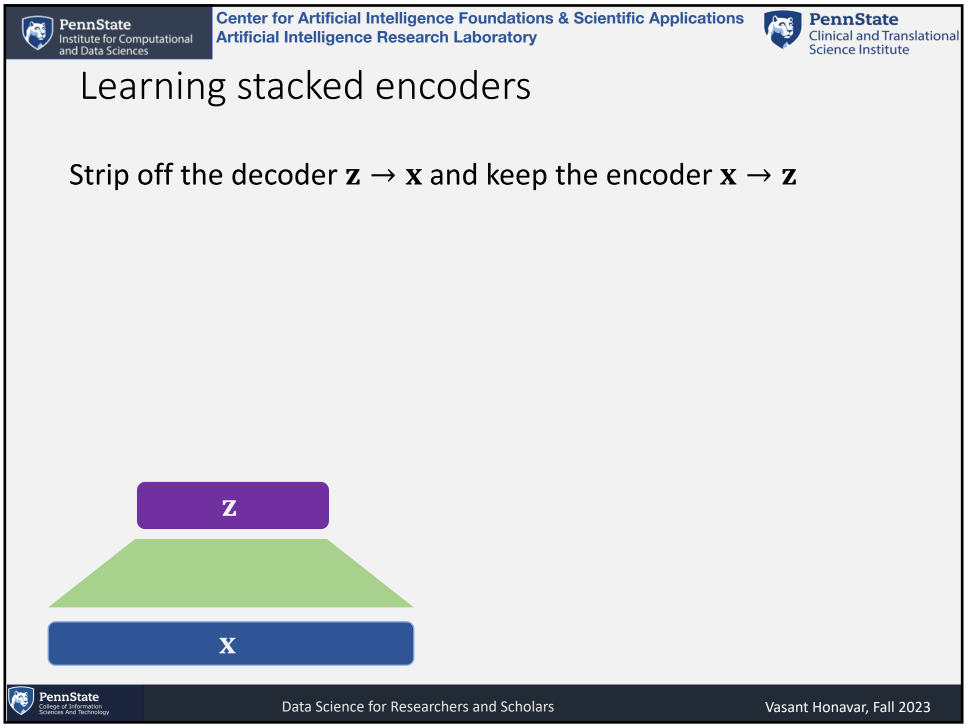
Stacked encoders

- Learn a multi-layer encoder one layer at a time
  - First learn an autoencoder $\mathbf{x} \rightarrow \boldsymbol{z} \rightarrow \mathbf{x}$
  - Strip off the decoder $\boldsymbol{z} \rightarrow \mathbf{x}$ and keep $\mathbf{x} \rightarrow \mathbf{z}$
  - Now learn an autoencoder $\mathbf{z} \rightarrow \mathbf{u} \rightarrow \mathbf{z}$
  - Strip off the decoder $\mathbf{u} \rightarrow \mathbf{z}$ and keep $\mathbf{z} \rightarrow \mathbf{u}$
  - Stack the encoders $\mathbf{x} \rightarrow \mathbf{z}$ and $\mathbf{z} \rightarrow \mathbf{u}$ to obtain the stacked encoder $\mathbf{x} \rightarrow \boldsymbol{z} \rightarrow \mathbf{u}$
  - Repeat the preceding steps if more layers are desired



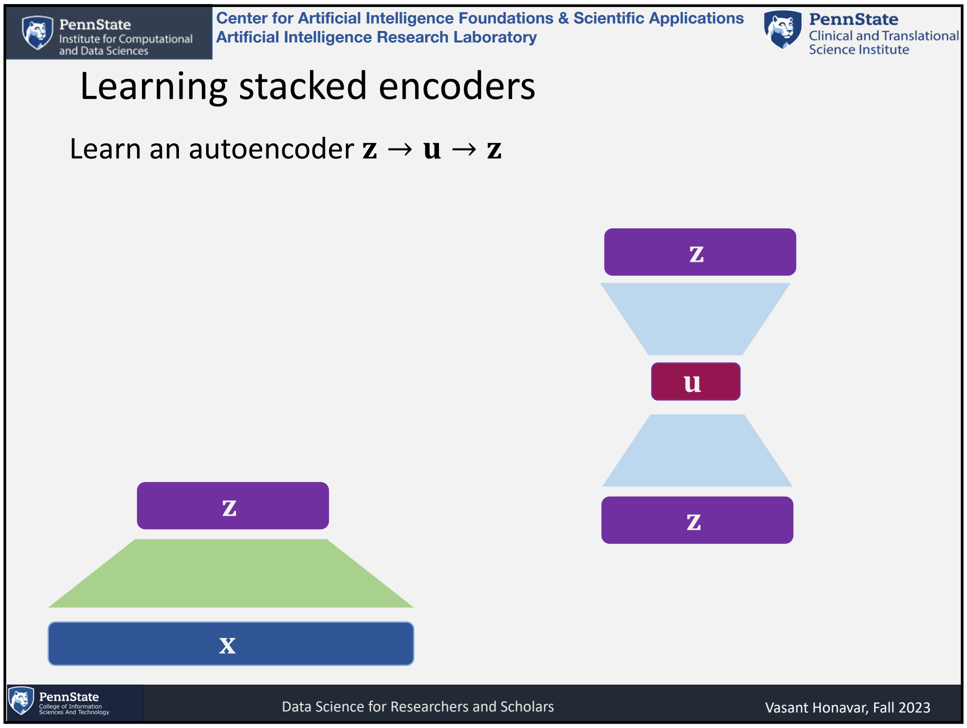
Learning stacked encoders

First learn an autoencoder $\mathbf{x} \rightarrow \boldsymbol{z} \rightarrow \mathbf{x}$



Learning stacked encoders

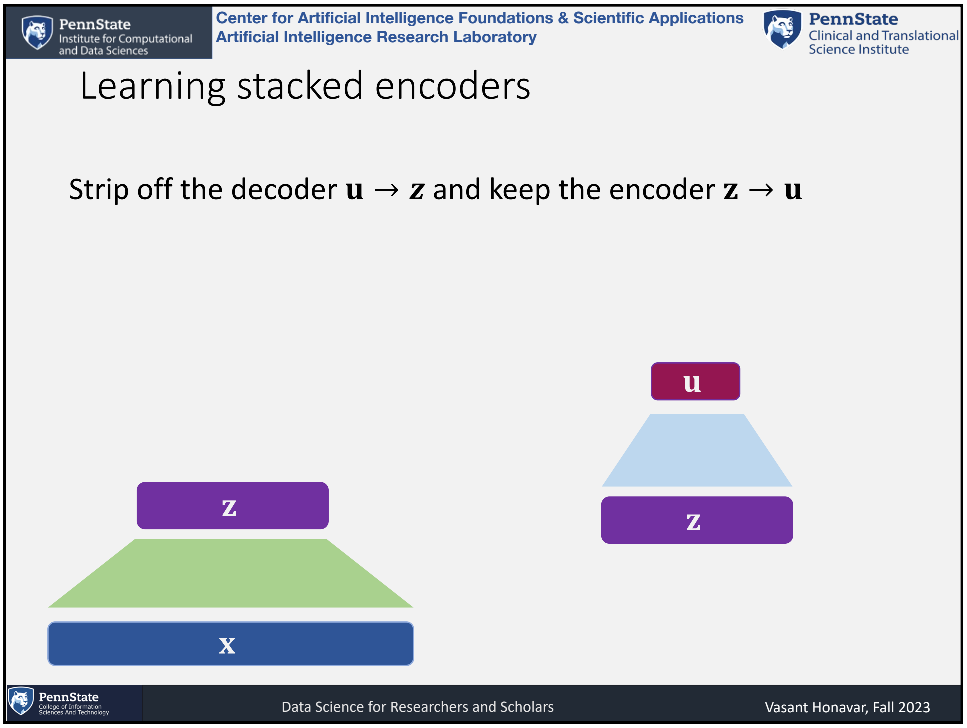
Strip off the decoder $\mathbf{z} \rightarrow \mathbf{x}$ and keep the encoder $\mathbf{x} \rightarrow \mathbf{z}$



Learning stacked encoders

Learn an autoencoder $\mathbf{z} \rightarrow \mathbf{u} \rightarrow \mathbf{z}$

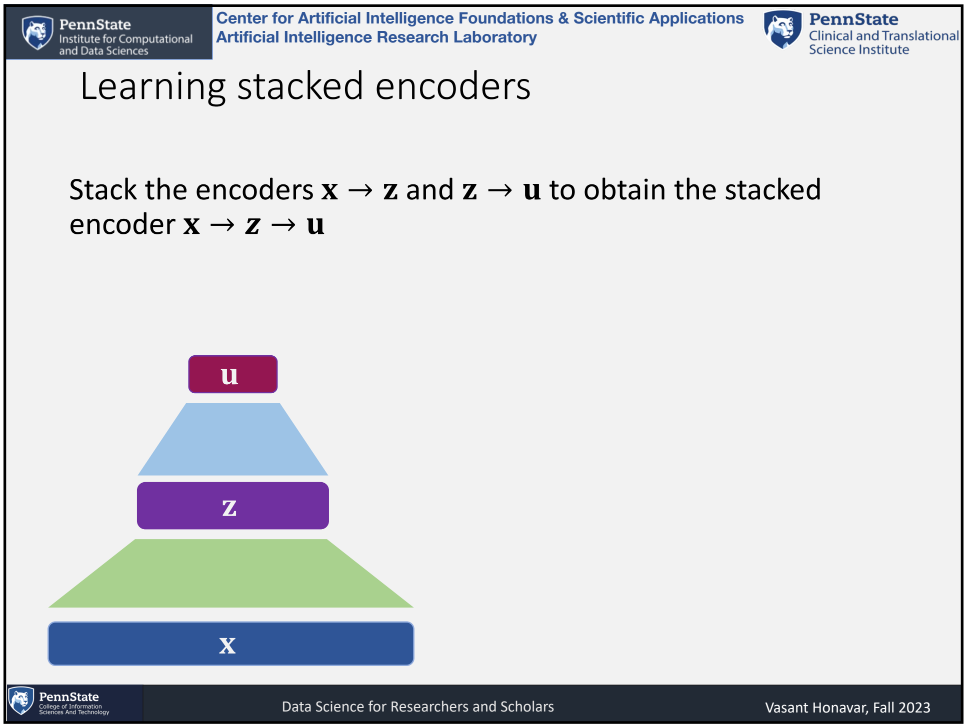
28



Learning stacked encoders

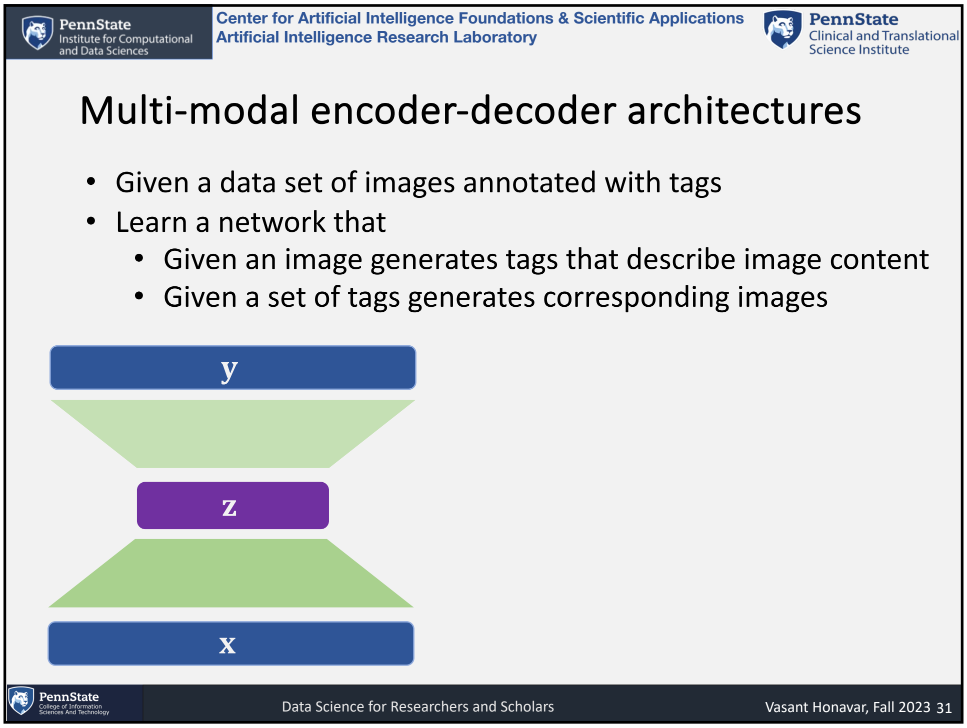
Strip off the decoder $\mathbf{u} \rightarrow \boldsymbol{z}$ and keep the encoder $\mathbf{z} \rightarrow \mathbf{u}$

29



Learning stacked encoders

Stack the encoders $\mathbf{x} \rightarrow \mathbf{z}$ and $\mathbf{z} \rightarrow \mathbf{u}$ to obtain the stacked encoder $\mathbf{x} \rightarrow \boldsymbol{z} \rightarrow \mathbf{u}$



Multi-modal encoder-decoder architectures

- Given a data set of images annotated with tags
- Learn a network that
  - Given an image generates tags that describe image content
  - Given a set of tags generates corresponding images

Autoencoders

- Autoencoders are only able to compress or extract useful features from data that are similar to training data
- The decoded output will be a noisy reconstruction of input
- Can be trained in an unsupervised fashion on large data sets
- The resulting encoding can be used to extract useful features from for supervised training of classifiers from much smaller data sets
- Examples of successful applications of autoencoders
  - Foundation models for computer vision

Denoising autoencoders

- The basic autoencoder minimizes the loss between $\mathbf{x}$ and the reconstruction $g\big(f(\mathbf{x})\big)$ where f is the encoder and g the decoder.
- Denoising autoencoders minimizes the loss between $\mathbf{x}$ and $g\big(f(\mathbf{x}+\mathbf{w})\big)$, where $\mathbf{w}$ is gaussian random noise
- Input and output of a denoising autoencoder

- Noise added to the input forces a denoising autoencoder to learn a mapping from noise perturbed training data to noise-free training data

ps://blog.keras.io/building-a

Sparse autoencoders

- Add a term to the loss function that forces sparse encoding of training data, e.g., sum of absolute values of the activations of the nodes in the hidden layer
- As a result, different subsets of the hidden nodes are activated by different inputs
- This is often combined by penalties for large weights (e.g., the square of the norm of the weight vectors).

Contractive autoencoders

- Contractive autoencoders make the *feature extraction function* (ie. encoder) robust in the presence of small perturbations of the input
- How?
- Instead of minimizing the loss, minimize the loss plus a term proportional to the magnitude of $\nabla_{\mathbf{x}} E$ i.e., the gradient of the loss with respect to the input $\mathbf{x}$

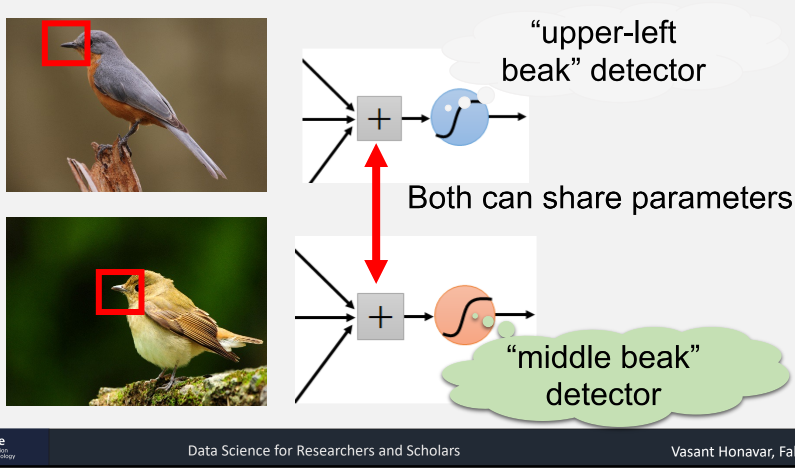
Stochastic Encoders and Decoders

- Modern autoencoders use stochastic mappings
- We can generalize the notion of the encoding and decoding functions to encoding and decoding distributions
- The resulting encoders are called variational autoencoders

Deep Convolutional networks

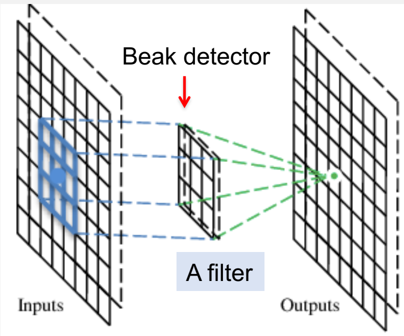
- Images have features at different spatial scales
- Need to be able to recognize objects regardless of where they appear in an image
- How can we modify neural networks to accommodate these considerations?
- Extract local features from images

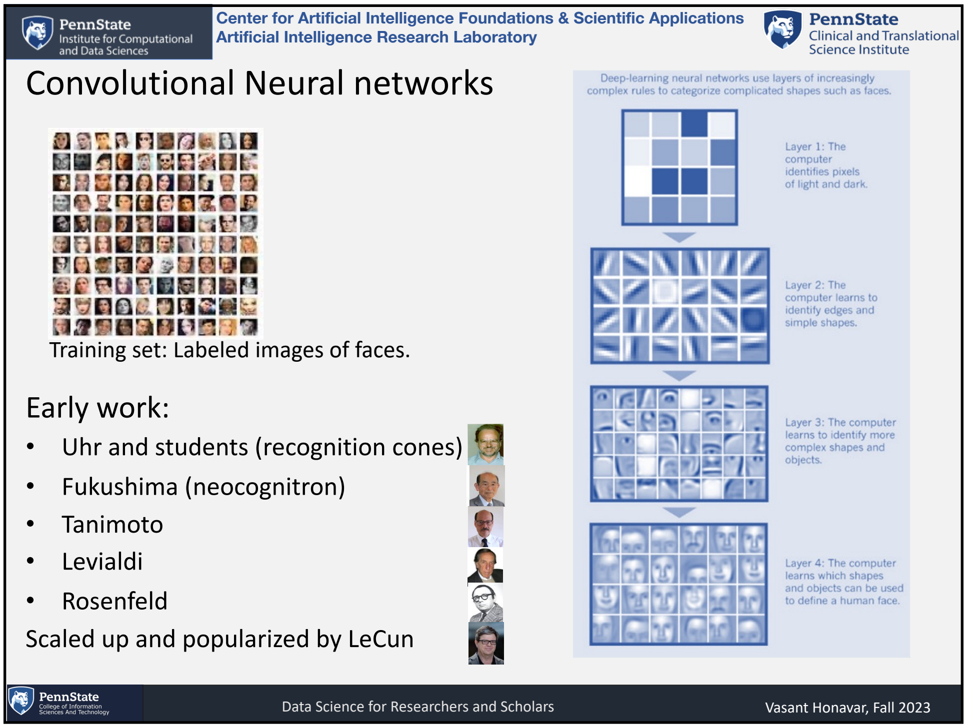
Translation invariance



Solution: convolutional neural network

- A CNN is a neural network with one or more convolutional layers
- A convolutional layer performs convolutional operations





Convolutional Neural networks

Training set: Labeled images of faces.

Early work:

- Uhr and students (recognition cones)
- Fukushima (neocognitron)
- Tanimoto
- Levialdi
- Rosenfeld

Scaled up and popularized by LeCun

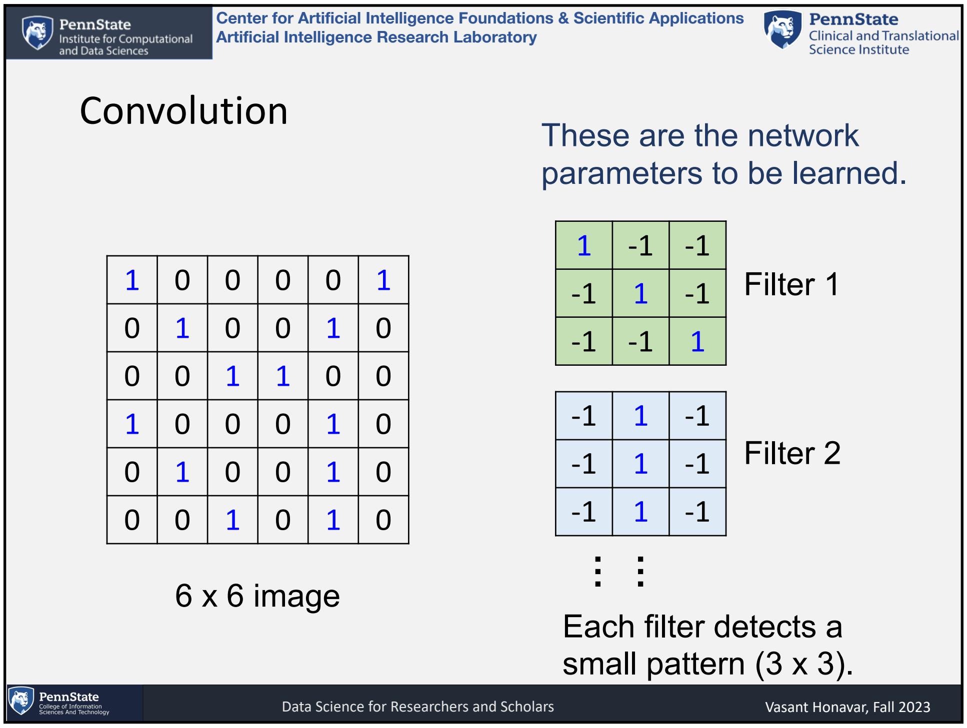
Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.

Layer 1: The computer identifies pixels of light and dark.

Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.



Convolution

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

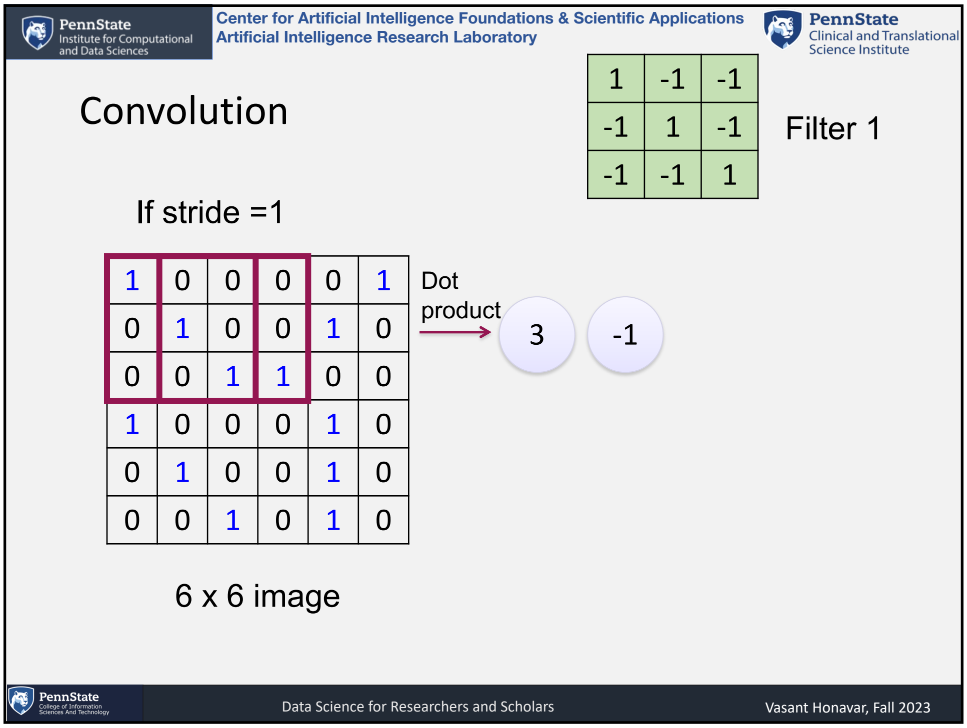
Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride =1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product

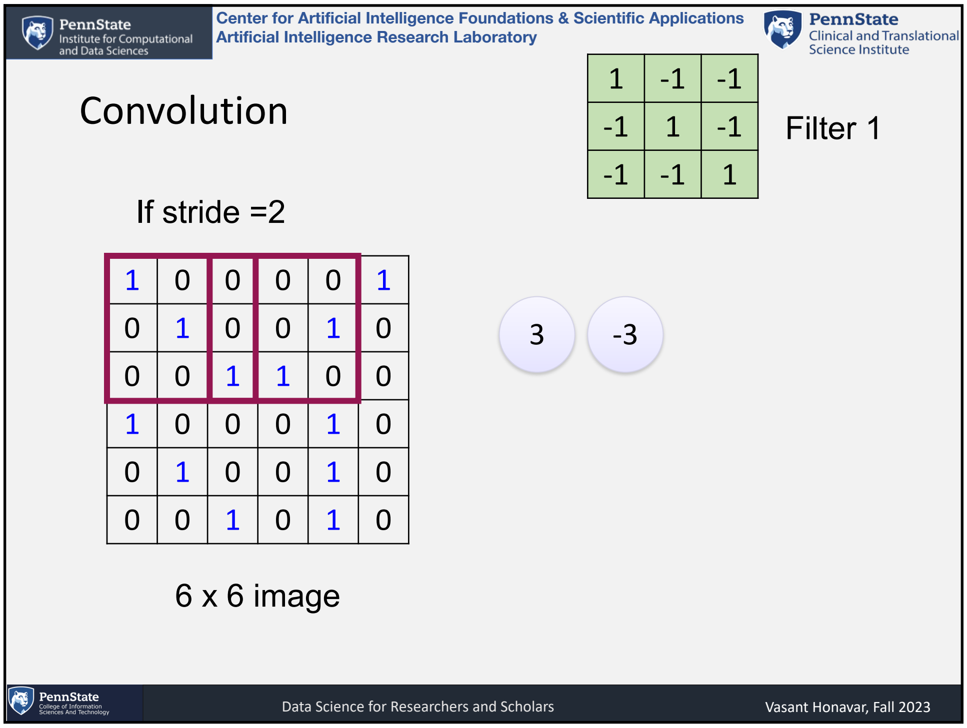
3 -1

6 x 6 image

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride =2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

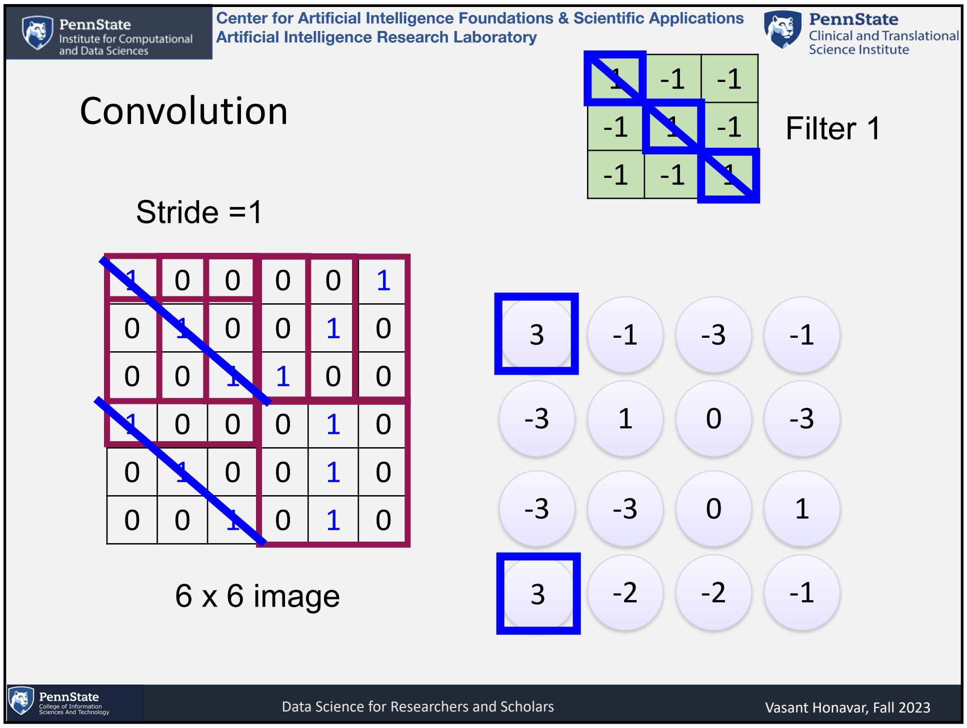
3 -3

6 x 6 image

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023



Convolution

Filter 1

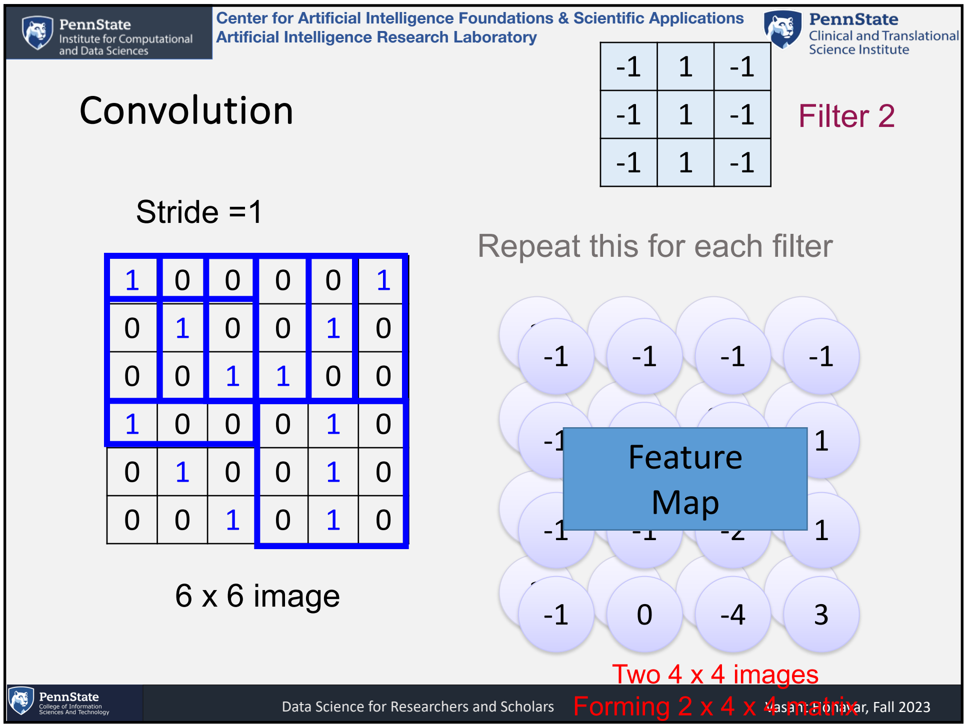
1	-1	-1
-1	1	-1
-1	-1	1

Stride =1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1



Convolution

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

Stride =1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

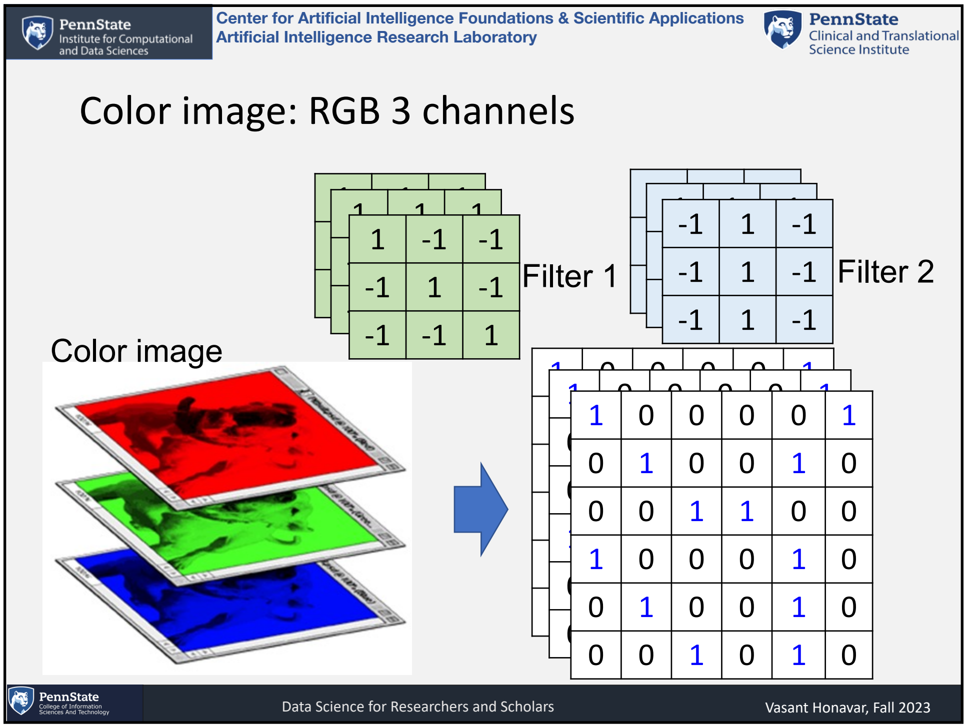
6 x 6 image

Repeat this for each filter

-1	-1	-1	-1
-1			1
-1	-1	-2	1
-1	0	-4	3

Feature Map

Two 4 x 4 images
Forming 2 x 4 x 4 matrix



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Color image: RGB 3 channels

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

Color image

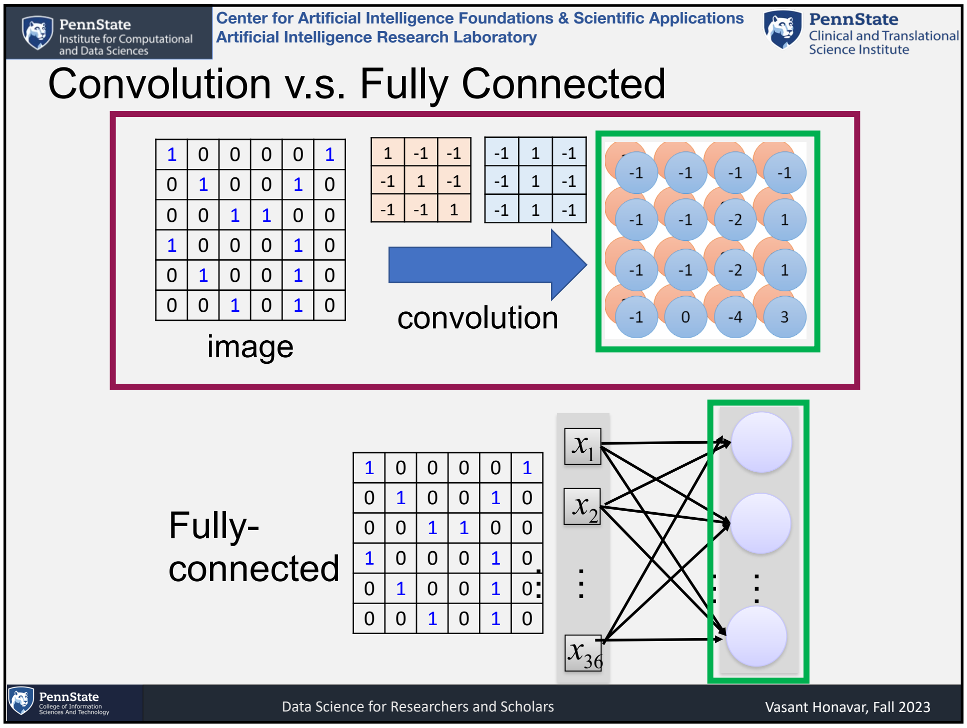
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023

46



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

Convolution v.s. Fully Connected

image

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

1	-1	-1
-1	1	-1
-1	-1	1

-1	1	-1
-1	1	-1
-1	1	-1

convolution

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Fully-connected

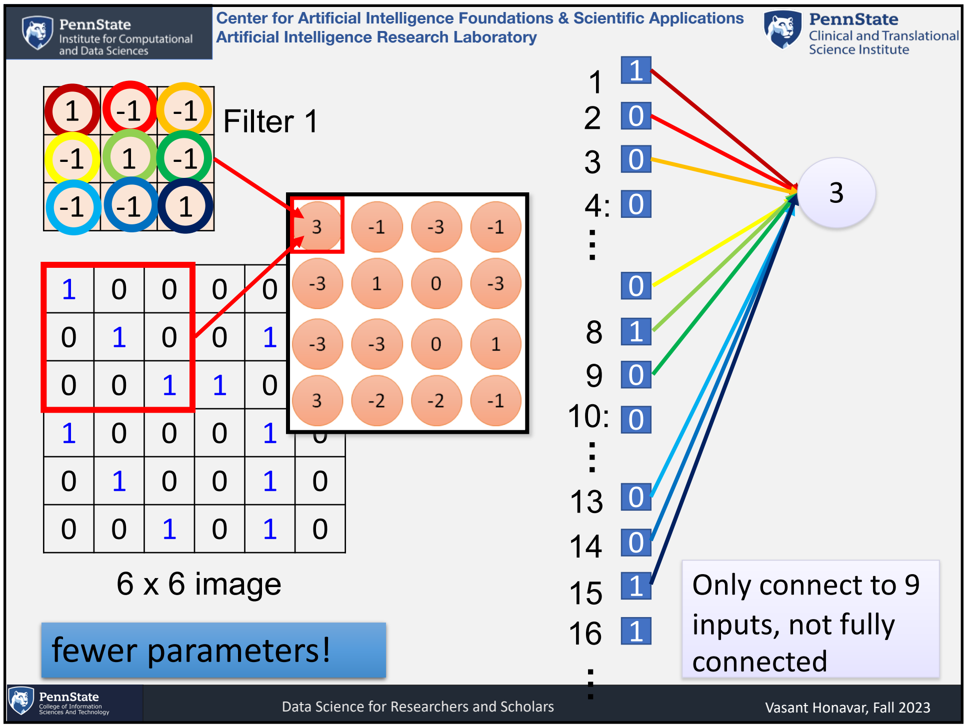
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

PennState College of Information Sciences And Technology

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023

47



Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

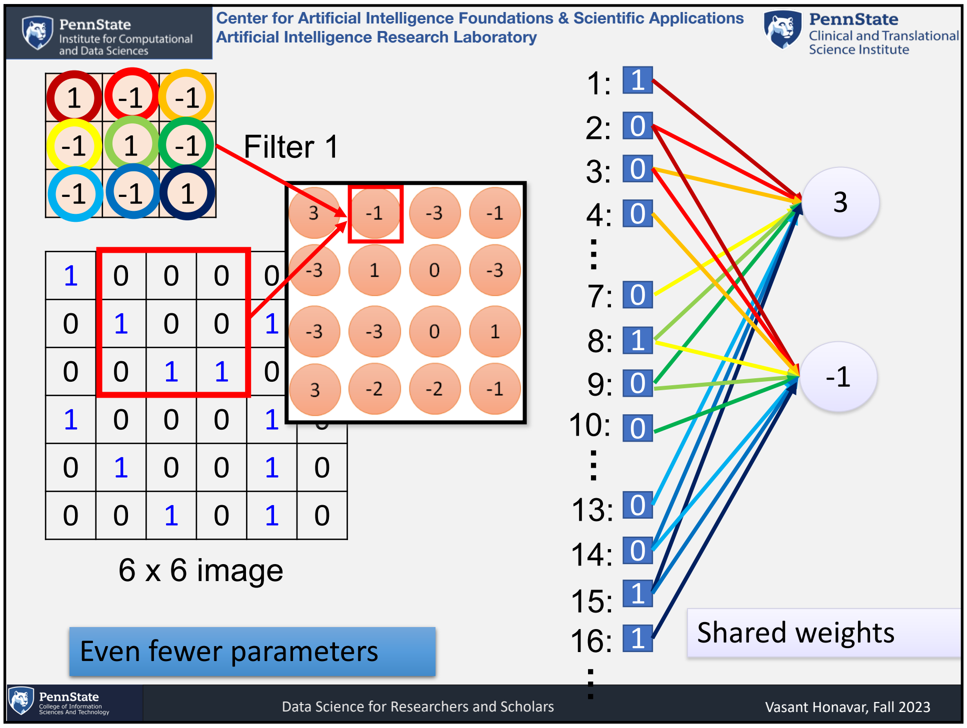
Only connect to 9 inputs, not fully connected

fewer parameters!

Data Science for Researchers and Scholars

Vasant Honavar, Fall 2023

48



Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Shared weights

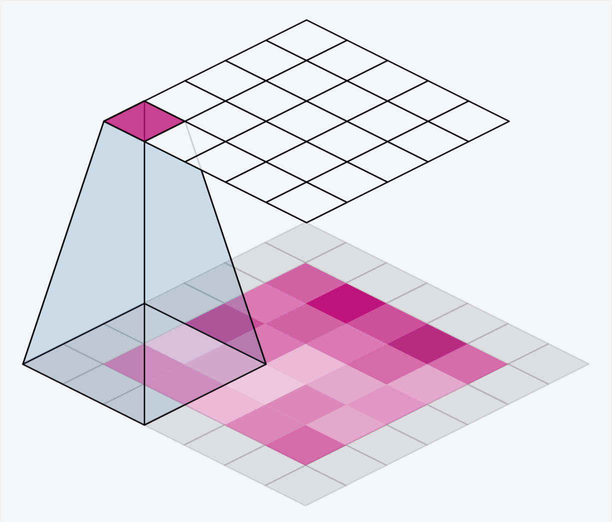
Even fewer parameters

Data Science for Researchers and Scholars

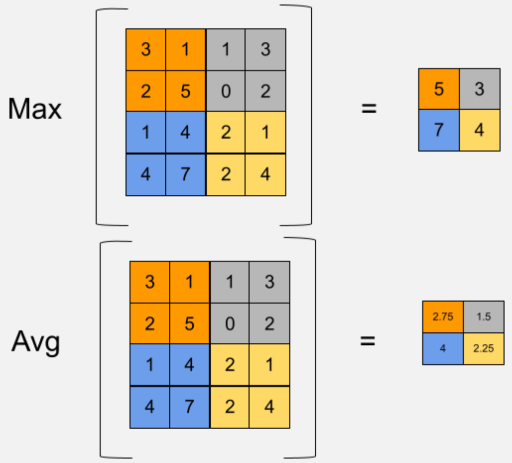
Vasant Honavar, Fall 2023

49

Convolution at work



Pooling



Max

3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

=

5	3
7	4

Avg

3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

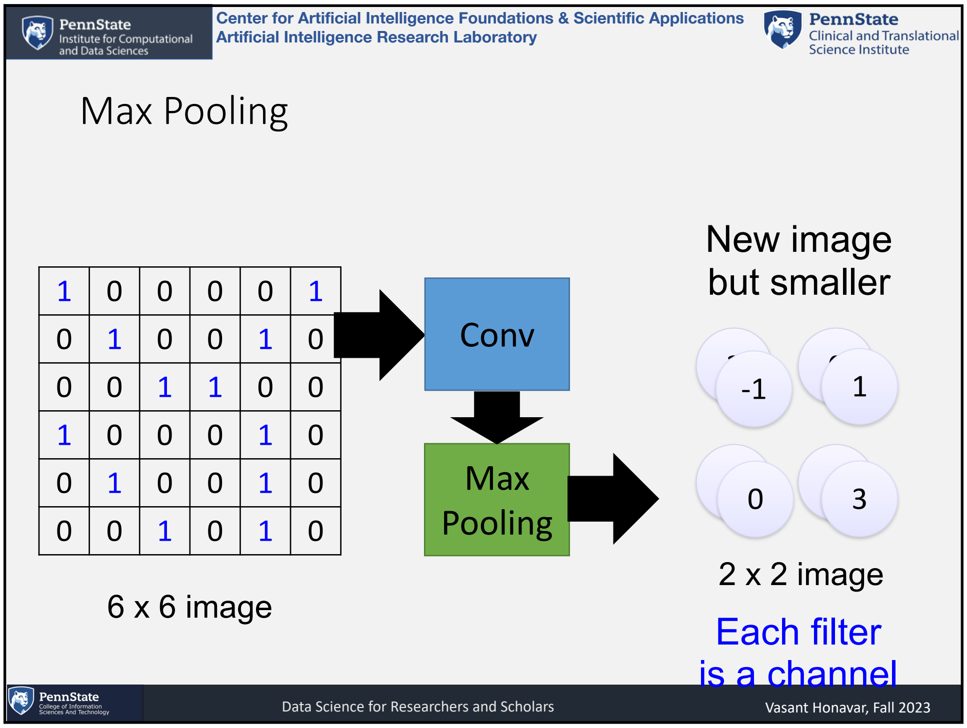
=

2.75	1.5
4	2.25

Pooling

- Down-samples the feature maps
- Reduces resolution
- Summarizes the features in a region
- Reduces the number of parameters to be learned
- Combines simpler features into more complex ones

Pooling can be traced back to early work on computer vision, e.g., recognition cones (Uhr, Li, Honavar), pyramids (Rosenfeld, Dyer, Tanimoto, Tsotsos)



Max Pooling

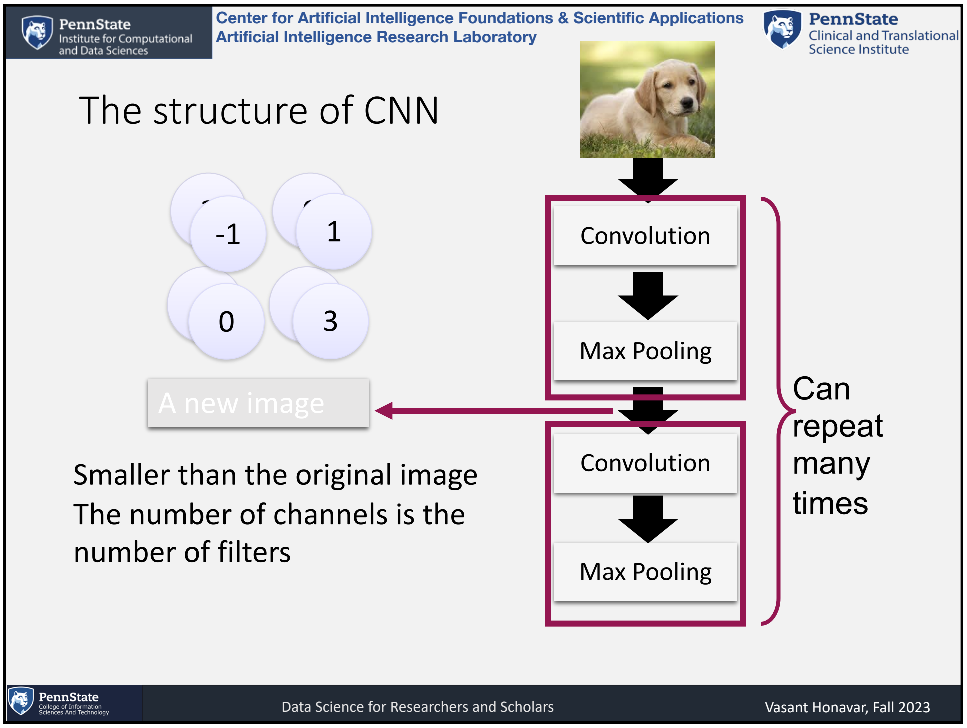
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

New image but smaller

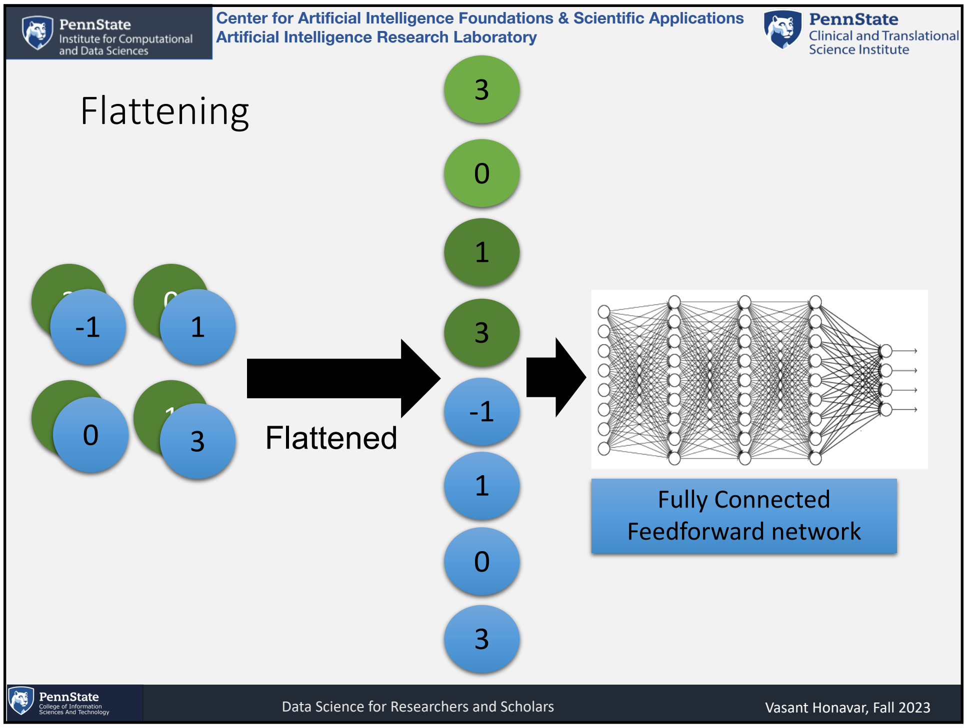
2 x 2 image

Each filter is a channel



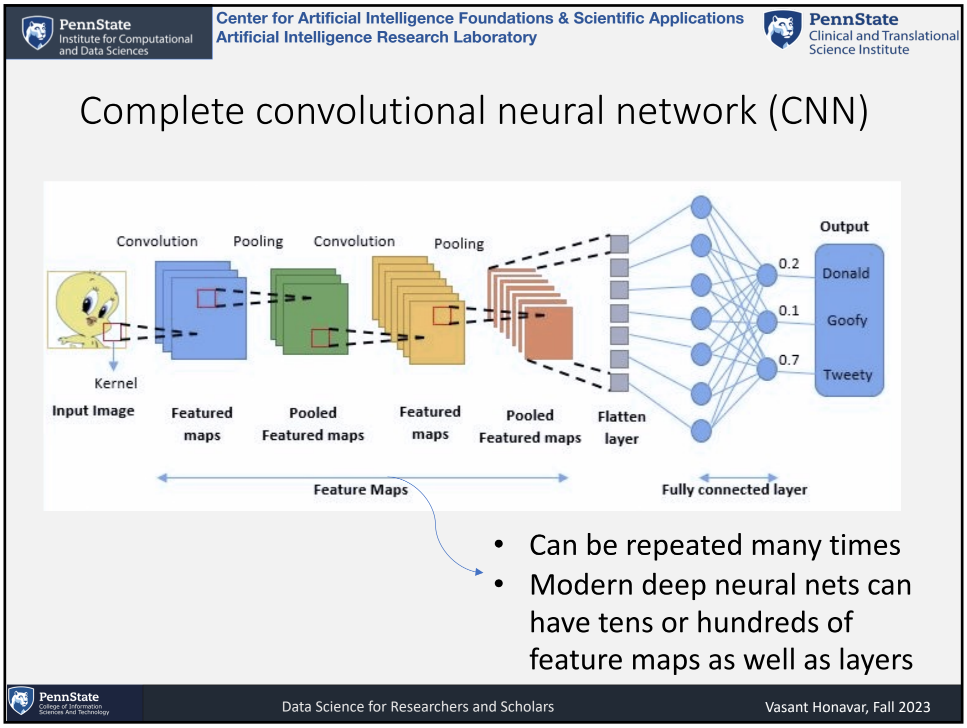
The structure of CNN

Smaller than the original image
The number of channels is the number of filters



Flattening

Fully Connected Feedforward network



Complete convolutional neural network (CNN)

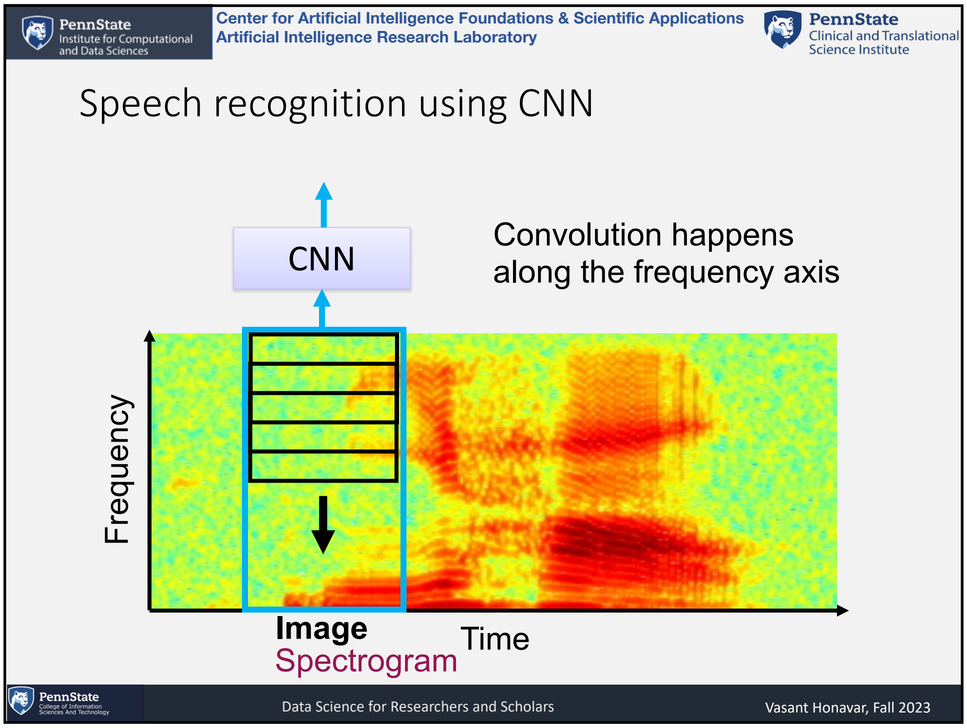
- Can be repeated many times
- Modern deep neural nets can have tens or hundreds of feature maps as well as layers

A CNN compresses a fully connected network

- Reduces the number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

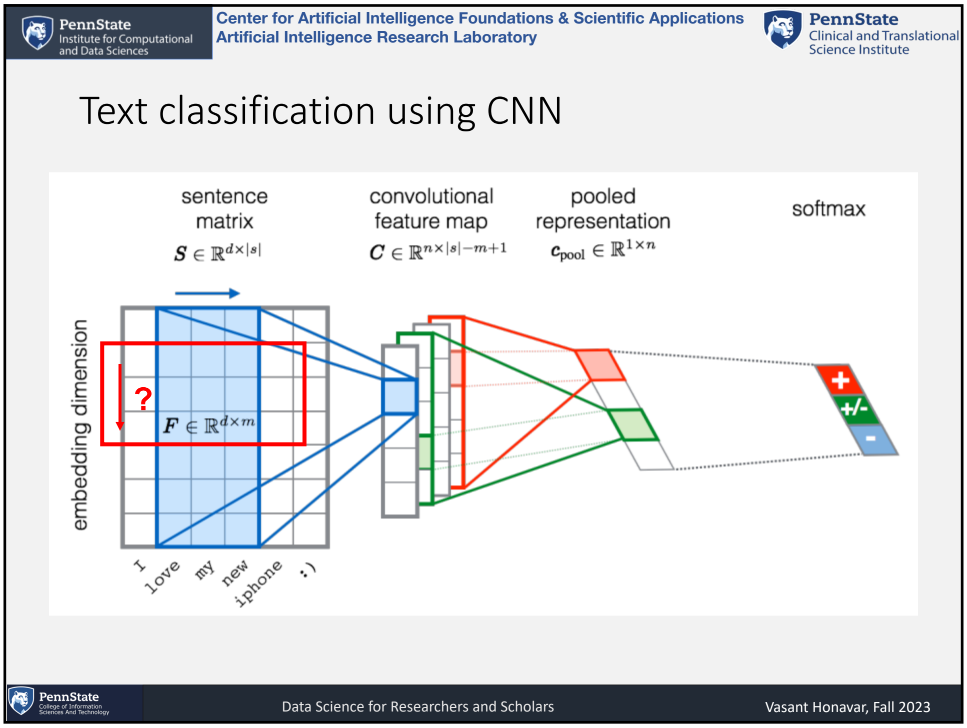
CNN Applications

- Image classification
- Speech recognition
- Biomolecular sequence classification
- Text classification



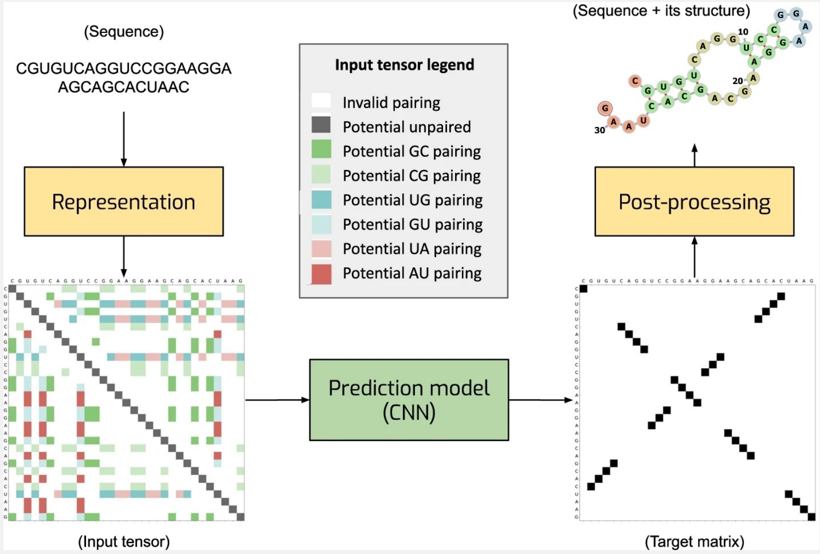
Speech recognition using CNN

Convolution happens along the frequency axis



Text classification using CNN

RNA secondary structure prediction using deep CNN



Recurrent neural networks

- Good for learning over sequences of data, e.g., a sentence of words
- LSTM (Long Short Term Memory) a popular architecture

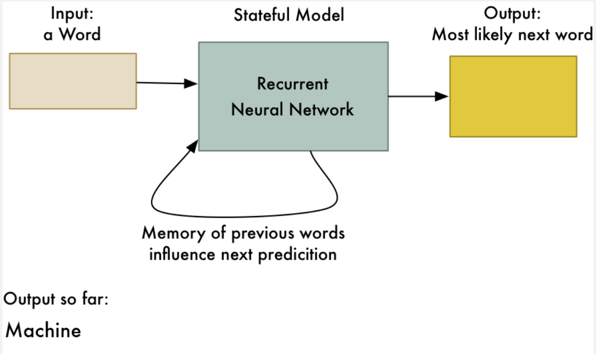
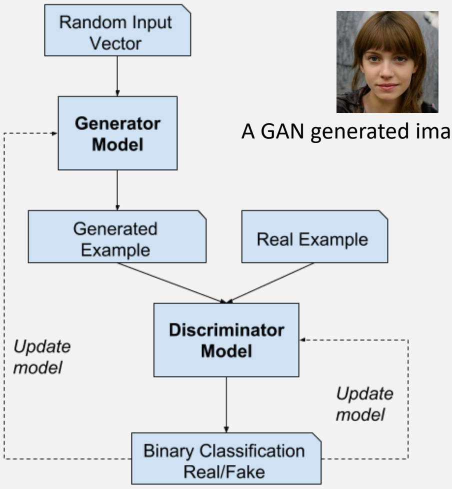


Image source: Adam Geitgey

Generative adversarial networks (GAN)

- System of two neural networks generator and discriminator compete against each other in a zero-sum game
  - Generator attempts to fool the discriminator
  - Discriminator attempts to defeat the generator
- Provides a kind of unsupervised learning that improves the network
- The result is a generative model whose output is indistinguishable from real data

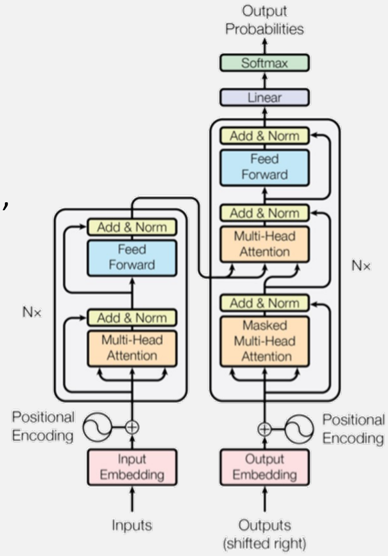


A GAN generated image

Goodfellow, et a.(2014). Generative Adversarial Networks. NIPS 2014. pp. 2672–2680.
Special case of an earlier model proposed by Jurgen Schmidhuber in 1991

Transformers

- Initially introduced for natural language processing (NLP)
- ”Transform” input text into output text
  - E.g., translation, text summarization, question answering
- Uses encoder-decoder architecture
- Since extended for text to image, sequence to structure, and many other applications
- Popular pretrained models available, e.g. BERT (encoder only) and GPT (decoder only)



Vaswani, A. et al. (2017). Attention is all you need. *In NIPS 2017.*

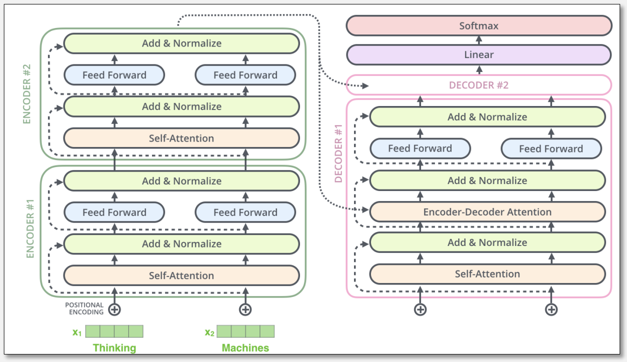
Transformers compared to older models

- The RNN and LSTM neural models were designed to process language and perform tasks like classification, summarization, translation, and sentiment detection
    - RNN: Recurrent Neural Network
    - LSTM: Long Short Term Memory
- In both models, layers get the next input word and have access to some previous words, allowing it to use the word’s left context
- They used word embeddings where each word was encoded as a vector of 100-300 real numbers representing its meaning

Transformers compared to older models

- Transformers extend this to allow the network to process a word input given the context: words both to the left and right
- Transformers add additional features, like attention which identifies the important words in context
- And break the problem into two parts:
    - An encoder (e.g., BERT)
    - A decoder (e.g., GPT)

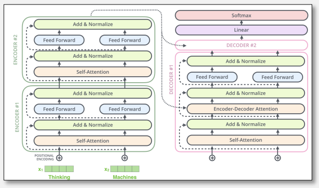
Transformer model



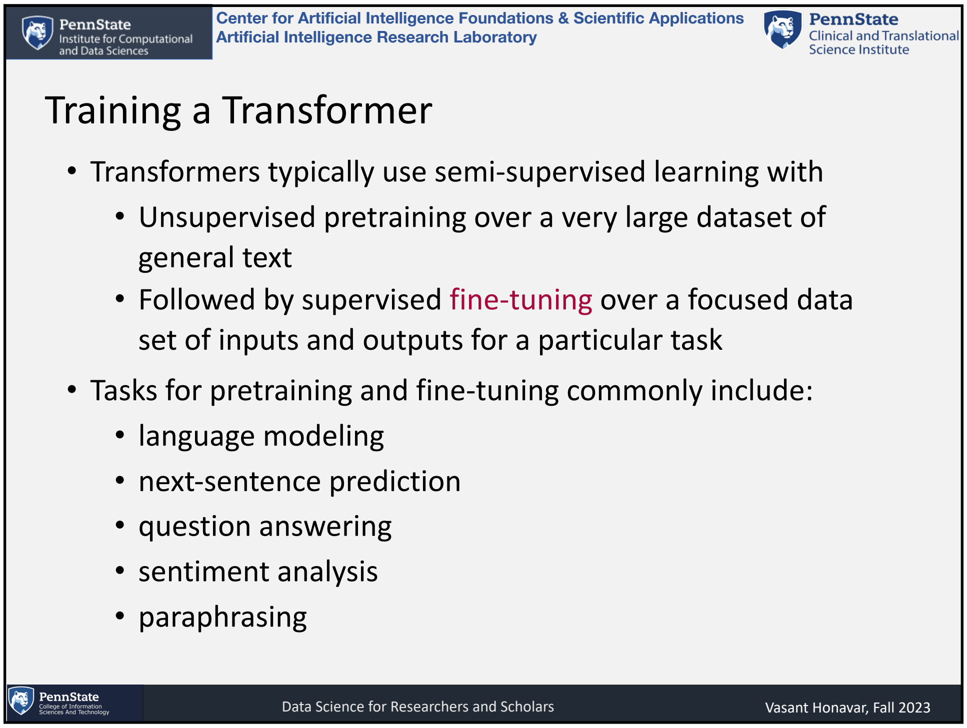
Encoder (e.g., BERT)

Decoder (e.g., GPT)

Transformers, GPT, and BERT

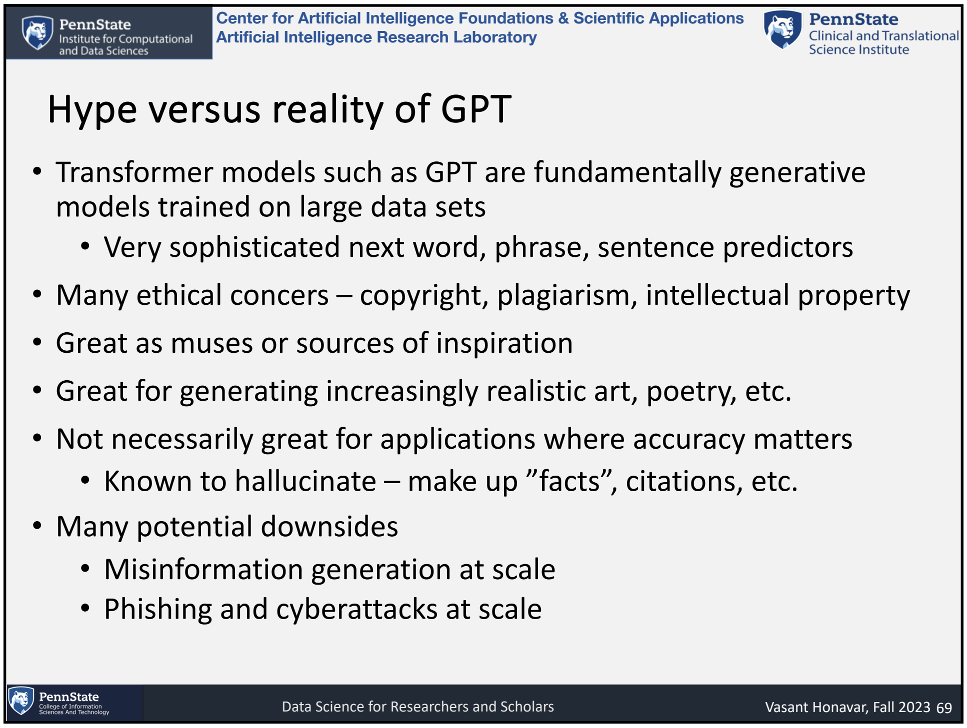


- A transformer uses an encoder stack to model input, and uses decoder stack to model output (using input information from encoder side)
- If we do not have input, we just want to model the “next word”, we can get rid of the encoder side of a transformer and output “next word” one by one. This gives us GPT
- If we are only interested in training a language model for the input for some other tasks, then we do not need the decoder of the transformer, that gives us BERT



Training a Transformer

- Transformers typically use semi-supervised learning with
  - Unsupervised pretraining over a very large dataset of general text
  - Followed by supervised fine-tuning over a focused data set of inputs and outputs for a particular task
- Tasks for pretraining and fine-tuning commonly include:
  - language modeling
  - next-sentence prediction
  - question answering
  - sentiment analysis
  - paraphrasing



Hype versus reality of GPT

- Transformer models such as GPT are fundamentally generative models trained on large data sets
  - Very sophisticated next word, phrase, sentence predictors
- Many ethical concers – copyright, plagiarism, intellectual property
- Great as muses or sources of inspiration
- Great for generating increasingly realistic art, poetry, etc.
- Not necessarily great for applications where accuracy matters
  - Known to hallucinate – make up ”facts”, citations, etc.
- Many potential downsides
  - Misinformation generation at scale
  - Phishing and cyberattacks at scale

Summary

- Unsupervised representation learning offers a powerful mechanism for learning useful features from complex data
- Multi-layer deep networks, given sufficient data, can be trained to perform many tasks that were once beyond the reach of machine learning
- Powerful tools Keras, Pytorch allow rapid prototyping of deep learning solutions
- But deep learning, as it is practiced today, is an environmental nightmare – need better approaches
- Deep learning systems can be easily fooled by adversarial data samples - need learning methods that are robust to adversarial attacks
- Deep learning produces black boxes that are hard to understand and explain –need better tools to explain the results of deep learning
- Advanced deep learning models e.g., LLMs are susceptible to hallucination, catastrophic forgetting, and other problems