University of Idaho
Department of Computer Science
Coeur d'Alene

Lecture #24

Autoencoders

Garrett Wells
revised October 17, 2024

2024-10-17

Lecture #24

Table of Contents I

Table of Contents I

Table of Contents
Review
Unsupervised Learning
Principal Component Analysis
Sparse Autoencoders
Applications
Data Denoising
Restyling, Reconstruction, & Recoloring
Other Types of Autoencoders

Review

- Three paradigms of ML
- Neural Networks
- Layers, Nodes, Weights, & Bias
- Prediction ➞ Feedforward
- Update ➞ Backpropagation
- Other Considerations...
  - Learning Rate
  - Learning Optimizations
  - Others

Review

- Three paradigms of ML
- Neural Networks
- Layers, Nodes, Weights, & Bias
- Prediction ➞ Feedforward
- Update ➞ Backpropagation
- Other Considerations...
  - Learning Rate
  - Learning Optimizations
  - Others

Unsupervised Learning

- Supervised Learning
  - Inputs & Labels
  - Labels/predictions are (usually) far less complex than input data
  - Some blob of pixels ➞ car
  - Input data ➞ symbolic/abstract representation
  - “If I could use *x* inputs to predict *y* I could do…”

Unsupervised Learning
- Supervised Learning
  - Inputs & Labels
  - Labels/predictions are (usually) far less complex than input data
  - Some blob of pixels ➞ car
  - Input data ➞ symbolic/abstract representation
  - “If I could use *x* inputs to predict *y* I could do…”

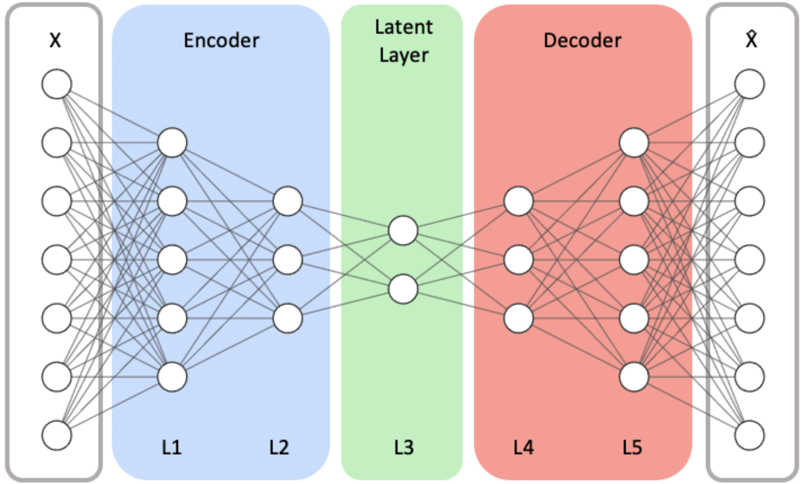
Unsupervised Learning

- Unsupervised Learning
  - Creative tasks
    - Symbols → complex data output
    - Set of facial features → image of human face
  - Data compression & Reconstruction
    - Music, Images, *etc.* have lossy data storage types
    - Algorithmic compression leaves artifacts
    - Adding color back into images

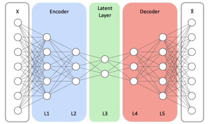
Unsupervised Learning
- Unsupervised Learning
  - Creative tasks
    - Symbols → complex data output
    - Set of facial features → image of human face
  - Data compression & Reconstruction
    - Music, Images, etc. have lossy data storage types
    - Algorithmic compression leaves artifacts
    - Adding color back into images

Autoencoders

Architecture. . .

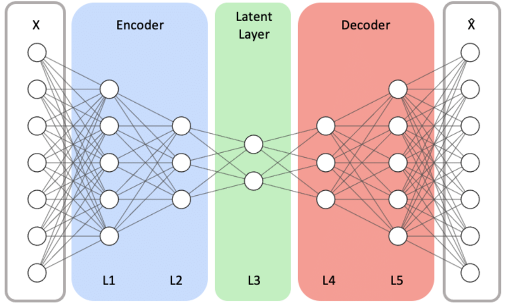


Autoencoders
Architecture...

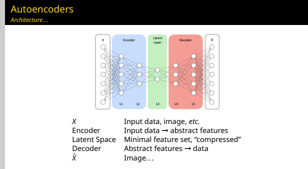


Autoencoders

Architecture. . .



X	Input data, image, *etc.*
Encoder	Input data ➞ abstract features
Latent Space	Minimal feature set, "compressed"
Decoder	Abstract features ➞ data
$\hat{X}$	Image. . .



Autoencoders

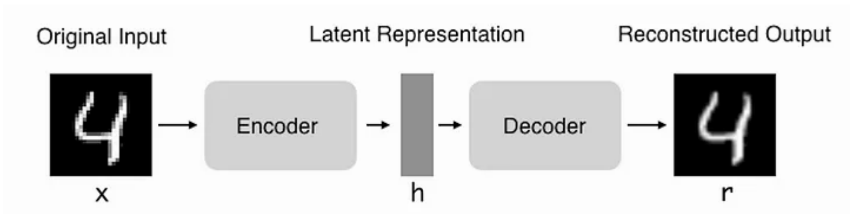


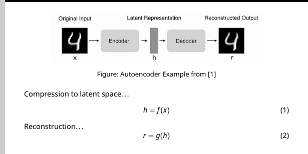
Figure: Autoencoder Example from [1]

Compression to latent space...

$$h = f(x) \tag{1}$$

Reconstruction...

$$r = g(h) \tag{2}$$



1. $r \approx x$, r(output) should approximate x(input) or match.

Autoencoders

Describing the latent space...

- Latent layer is the narrowest

Autoencoders
Describing the latent space...
- Latent layer is the narrowest

Autoencoders

Describing the latent space...

- Latent layer is the narrowest
- A "complete" layer has the same number of nodes as inputs

Autoencoders
Describing the latent space...
- Latent layer is the narrowest
- A "complete" layer has the same number of nodes as inputs

Autoencoders

Describing the latent space. . .

- Latent layer is the narrowest
- A “complete” layer has the same number of nodes as inputs
- Thus we describe the latent space as being “undercomplete”

Autoencoders
Describing the latent space. . .
- Latent layer is the narrowest
- A “complete” layer has the same number of nodes as inputs
- Thus we describe the latent space as being “undercomplete”

Autoencoders

Describing the latent space…

- Latent layer is the narrowest
- A “complete” layer has the same number of nodes as inputs
- Thus we describe the latent space as being “undercomplete”
- Layer with more nodes than inputs ☞ “overcomplete”
- If complete, not learning useful characteristics

Autoencoders
Describing the latent space…
- Latent layer is the narrowest
- A “complete” layer has the same number of nodes as inputs
- Thus we describe the latent space as being “undercomplete”
- Layer with more nodes than inputs ☞ “overcomplete”
- If complete, not learning useful characteristics

Autoencoders

Describing the latent space…

- Latent layer is the narrowest
- A “complete” layer has the same number of nodes as inputs
- Thus we describe the latent space as being “undercomplete”
- Layer with more nodes than inputs ☞ “overcomplete”
- If complete, not learning useful characteristics
- Just copying data

Autoencoders
Describing the latent space…
- Latent layer is the narrowest
- A “complete” layer has the same number of nodes as inputs
- Thus we describe the latent space as being “undercomplete”
- Layer with more nodes than inputs ☞ “overcomplete”
- If complete, not learning useful characteristics
- Just copying data

The Ideal Autoencoder[2]…

1. Sensitive enough to the inputs to reconstruct the input.
2. Insensitive enough to avoid memorizing the training data.

Autoencoders
Art & Science…

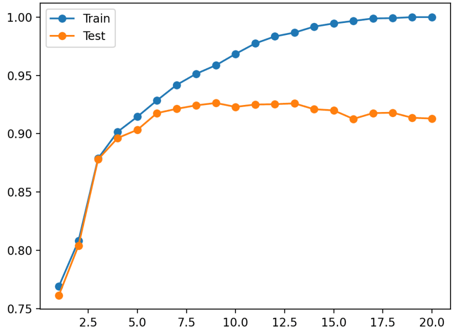
The Ideal Autoencoder[2]…

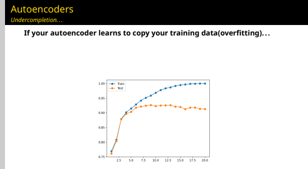
1. Sensitive enough to the inputs to reconstruct the input.
2. Insensitive enough to avoid memorizing the training data.

Autoencoders

Undercompletion…

If your autoencoder learns to copy your training data(overfitting)…





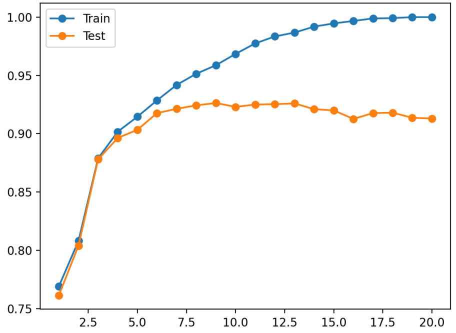
1. Note that this graph is more typical of what we would see in supervised learning where calculating performance and accuracy is more straight forward.
2. This is why true model performance is determined by accuracy on data outside the training data.

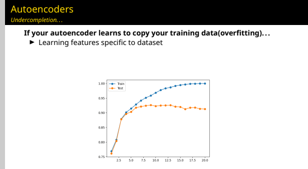
Autoencoders

Undercompletion...

If your autoencoder learns to copy your training data(overfitting)...

- Learning features specific to dataset





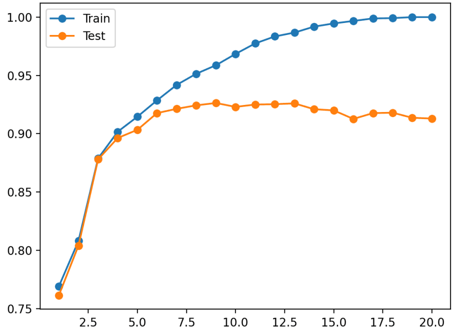
1. Note that this graph is more typical of what we would see in supervised learning where calculating performance and accuracy is more straight forward.
2. This is why true model performance is determined by accuracy on data outside the training data.

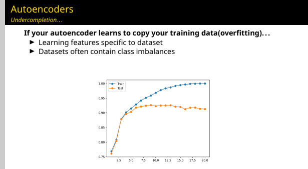
Autoencoders

Undercompletion...

If your autoencoder learns to copy your training data(overfitting)...

- Learning features specific to dataset
- Datasets often contain class imbalances





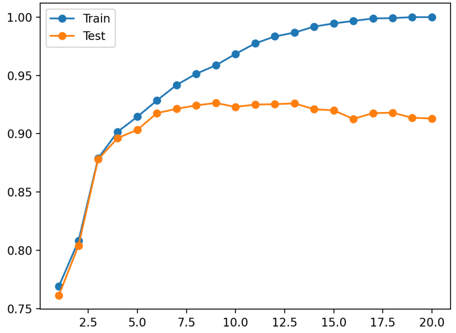
1. Note that this graph is more typical of what we would see in supervised learning where calculating performance and accuracy is more straight forward.
2. This is why true model performance is determined by accuracy on data outside the training data.

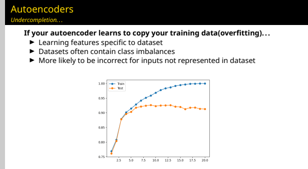
Autoencoders

Undercompletion...

If your autoencoder learns to copy your training data(overfitting)...

- Learning features specific to dataset
- Datasets often contain class imbalances
- More likely to be incorrect for inputs not represented in dataset





1. Note that this graph is more typical of what we would see in supervised learning where calculating performance and accuracy is more straight forward.
2. This is why true model performance is determined by accuracy on data outside the training data.

Principal Component Analysis

Dimensionality reduction...

Principal Component Analysis

Principal component analysis is a dimensionality reduction and machine learning method used to simplify a large dataset into a smaller set while still maintaining significant patterns and trends.

Principal Component Analysis
Dimensionality reduction...
Principal Component Analysis
Principal component analysis is a dimensionality reduction and machine learning method used to simplify a large dataset into a smaller set while still maintaining significant patterns and trends.

Principal Component Analysis

Dimensionality reduction…

Principal Component Analysis

Principal component analysis is a **dimensionality reduction** and machine learning method used to simplify a large dataset into a smaller set while still maintaining significant patterns and trends.

Principal Component Analysis
Dimensionality reduction…
Principal Component Analysis
Principal component analysis is a **dimensionality reduction** and machine learning method used to simplify a large dataset into a smaller set while still maintaining significant patterns and trends.

Principal Component Analysis

Dimensionality reduction…

Principal Component Analysis

Principal component analysis is a dimensionality reduction and machine learning **method used to simplify a large dataset into a smaller set** while still maintaining significant patterns and trends.

Principal Component Analysis
Dimensionality reduction…
Principal Component Analysis
Principal component analysis is a dimensionality reduction and machine learning method used to simplify a large dataset into a smaller set while still maintaining significant patterns and trends.

Principal Component Analysis

Dimensionality reduction...

1. Standardize the range of continuous initial variables.

Principal Component Analysis
Dimensionality reduction...
1. Standardize the range of continuous initial variables.

Principal Component Analysis

Dimensionality reduction. . .

1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.

Principal Component Analysis
Dimensionality reduction...
1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.

Principal Component Analysis

Dimensionality reduction...

1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.

Principal Component Analysis
Dimensionality reduction...
1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.

Principal Component Analysis

Dimensionality reduction...

1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.
4. Create a feature vector to decide which principal components to keep.

Principal Component Analysis
Dimensionality reduction...
1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.
4. Create a feature vector to decide which principal components to keep.

Principal Component Analysis

Dimensionality reduction…

1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.
4. Create a feature vector to decide which principal components to keep.
5. Recast the data along principal component's axes.

Principal Component Analysis
Dimensionality reduction…
1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.
4. Create a feature vector to decide which principal components to keep.
5. Recast the data along principal component's axes.

Principal Component Analysis

Dimensionality reduction...

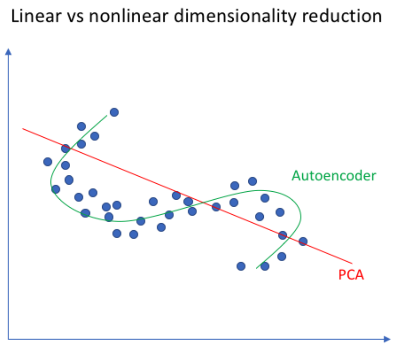
1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.
4. Create a feature vector to decide which principal components to keep.
5. Recast the data along principal component's axes.

Principal Component Analysis
Dimensionality reduction...
1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute eigenvectors and eigenvalues of the covariance matrix to identify principal components.
4. Create a feature vector to decide which principal components to keep.
5. Recast the data along principal component's axes.

Principal Component Analysis

Dimensionality reduction…

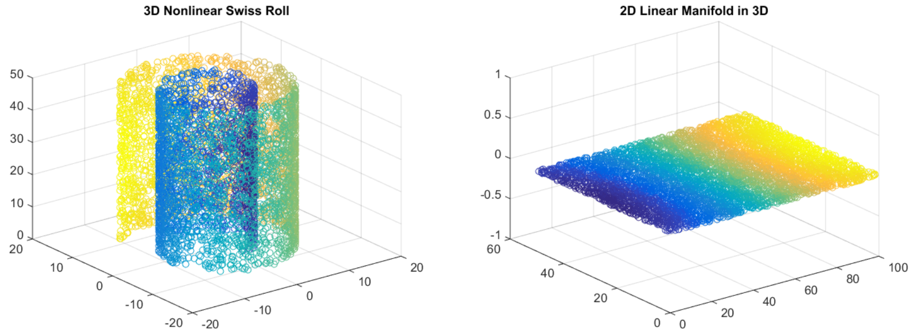
- PCA is trying to find a **lower dimensional hyperplane representation** of data[2].
- Autoencoders can learn **non-linear manifolds**.

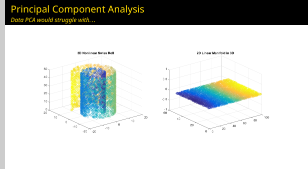


Principal Component Analysis
Dimensionality reduction…
PCA is trying to find a lower dimensional hyperplane representation of data[2].
Autoencoders can learn non-linear manifolds.

Principal Component Analysis

Data PCA would struggle with…





Sparse Autoencoders

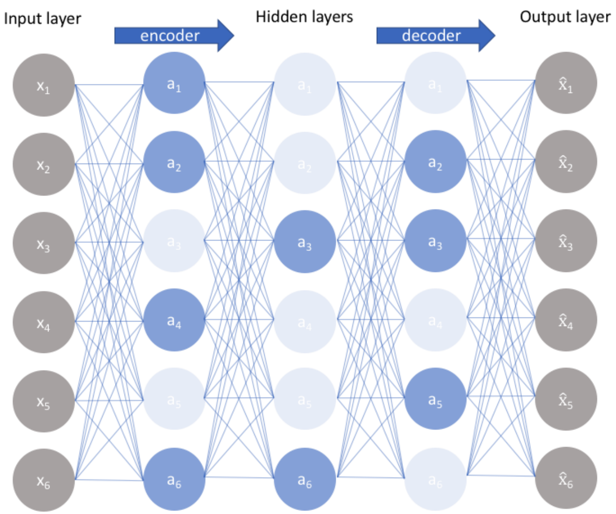
Instead of reducing the number of nodes in latent space, create a loss function penalizing node activations within the latent space.

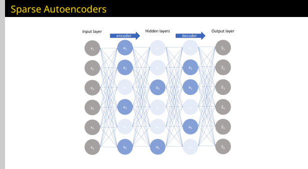
Sparse Autoencoders

Sparse Autoencoders

Instead of reducing the number of nodes in latent space, create a loss function penalizing node activations within the latent space.

Sparse Autoencoders





Sparse Autoencoders

Theoretical outcome...

1. Limit network's ability to memorize input data[2]
2. Network still able to extract features from input data

Sparse Autoencoders
Theoretical outcome...
1. Limit network's ability to memorize input data[2]
2. Network still able to extract features from input data

Activation Based Loss [2]

L1 Regularization:

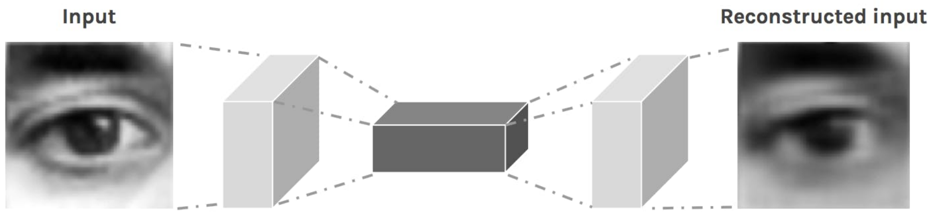
$$L(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}| \tag{1}$$

λ tuning parameter
a activations in layer h for observation i

Sparse Autoencoders
Activation Based Loss [2]
L1 Regularization:
$L(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}|$ (1)
λ tuning parameter
a activations in layer h for observation i

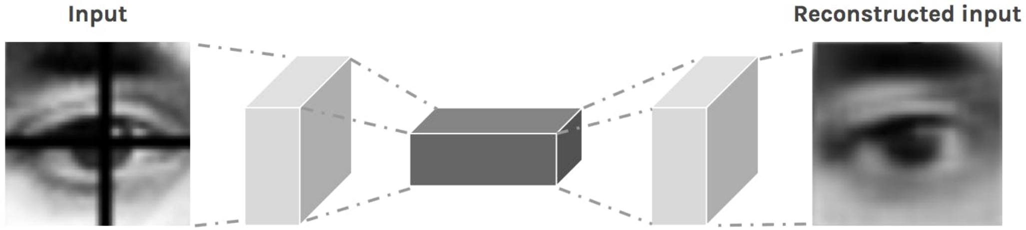
Data Denoising

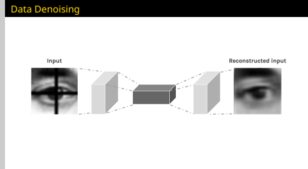
- Input Image → Autoencoder reconstructs image
- May lose some information, but good if loses noise



Data Denoising
► Input Image → Autoencoder reconstructs image
► May lose some information, but good if loses noise

Data Denoising





Restyling & Recoloring

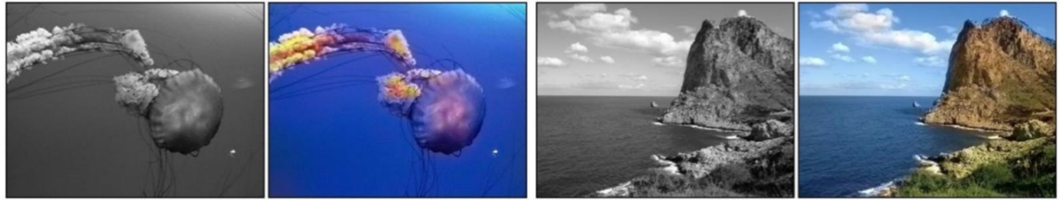


Figure: Example of Recoloring from [3]

- Grayscale image
- Score based on difference between color image and autoencoder output

Restyling & Recoloring

Figure: Example of Recoloring from [3]
- Grayscale image
- Score based on difference between color image and autoencoder output

Image Upscaling

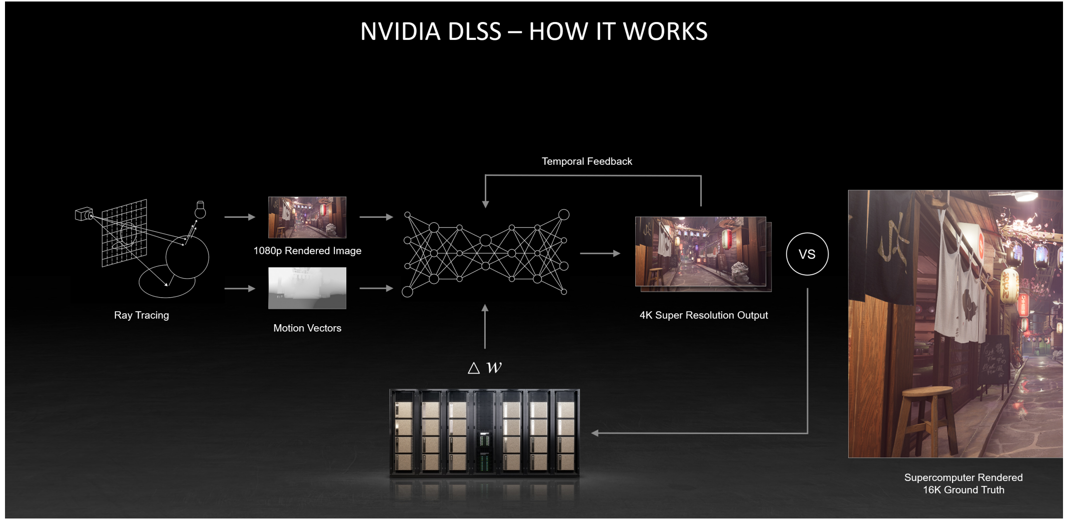
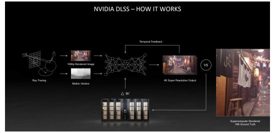


Image Upscaling



Types of Autoencoders

1. Vanilla/Basic [1]
2. Multilayer
3. Convolutional
4. Regularized

Types of Autoencoders

1. Vanilla/Basic [1]
2. Multilayer
3. Convolutional
4. Regularized

Vanilla

One hidden layer…

```
input_size = 784
hidden_size = 64
output_size = 784

x = Input(shape=(input_size,))

# Encoder
h = Dense(hidden_size, activation='relu')(x)

# Decoder
r = Dense(output_size, activation='sigmoid')(h)

autoencoder = Model(input=x, output=r)
autoencoder.compile(optimizer='adam', loss='mse')
```


Vanilla
One hidden layer…
input_size = 784
hidden_size = 64
output_size = 784
x = Input(shape=(input_size,))
# Encoder
h = Dense(hidden_size, activation='relu')(x)
# Decoder
r = Dense(output_size, activation='sigmoid')(h)
autoencoder = Model(input=x, output=r)
autoencoder.compile(optimizer='adam', loss='mse')

1. “Vanilla” is a model with the minimum number of layers. Not an example of “deep” learning.

Multilayer

Four hidden layers…

```
input_size = 784
hidden_size = 128
code_size = 64
x = Input(shape=(input_size,))

# Encoder
hidden_1 = Dense(hidden_size, activation='relu')(x)
h = Dense(code_size, activation='relu')(hidden_1)

# Decoder
hidden_2 = Dense(hidden_size, activation='relu')(h)
r = Dense(input_size, activation='sigmoid')(hidden_2)

autoencoder = Model(input=x, output=r)
autoencoder.compile(optimizer='adam', loss='mse')
```


Multilayer
Four hidden layers…
input_size = 784
hidden_size = 128
code_size = 64
x = Input(shape=(input_size,))
# Encoder
hidden_1 = Dense(hidden_size, activation='relu')(x)
h = Dense(code_size, activation='relu')(hidden_1)
# Decoder
hidden_2 = Dense(hidden_size, activation='relu')(h)
r = Dense(input_size, activation='sigmoid')(hidden_2)
autoencoder = Model(input=x, output=r)
autoencoder.compile(optimizer='adam', loss='mse')

1. Any network with any number of nodes.

Convolutional

Convolutional

```
# Encoder
conv1_1 = Conv2D(16, (3, 3),...)(x)
pool1 = MaxPooling2D((2, 2),...)(conv1_1)
conv1_2 = Conv2D(8, (3, 3),...)(pool1)
pool2 = MaxPooling2D((2, 2),...)(conv1_2)
conv1_3 = Conv2D(8, (3, 3),...)(pool2)
h = MaxPooling2D((2, 2),...)(conv1_3)
# Decoder
conv2_1 = Conv2D(8, (3, 3),...)(h)
up1 = UpSampling2D((2, 2))(conv2_1)
conv2_2 = Conv2D(8, (3, 3), ...)(up1)
up2 = UpSampling2D((2, 2))(conv2_2)
conv2_3 = Conv2D(16, (3, 3),...)(up2)
up3 = UpSampling2D((2, 2))(conv2_3)
r = Conv2D(1, (3, 3),...)(up3)
```


Convolutional
Convolutional
# Encoder
conv1_1 = Conv2D(16, (3, 3),...)(x)
pool1 = MaxPooling2D((2, 2),...)(conv1_1)
conv1_2 = Conv2D(8, (3, 3),...)(pool1)
pool2 = MaxPooling2D((2, 2),...)(conv1_2)
conv1_3 = Conv2D(8, (3, 3),...)(pool2)
h = MaxPooling2D((2, 2),...)(conv1_3)
# Decoder
conv2_1 = Conv2D(8, (3, 3),...)(h)
up1 = UpSampling2D((2, 2))(conv2_1)
conv2_2 = Conv2D(8, (3, 3), ...)(up1)
up2 = UpSampling2D((2, 2))(conv2_2)
conv2_3 = Conv2D(16, (3, 3),...)(up2)
up3 = UpSampling2D((2, 2))(conv2_3)
r = Conv2D(1, (3, 3),...)(up3)

Regularized Autoencoders

Sparse & Denoising

- Encompasses methods of constraining reconstruction
- Penalizing activations, *etc.*

```
h = Dense(hidden_size,
          activation='relu',
          activity_regularizer=regularizers.l1(10e-5))(x)
```


Regularized Autoencoders
Sparse & Denoising
► Encompasses methods of constraining reconstruction
► Penalizing activations, etc.
h = Dense(hidden_size,
activation='relu',
activity_regularizer=regularizers.l1(10e-5))(x)

1. These are differentiated by adding a different regularization method to the layers.

Bibliography I

[1] N. Hubens, “Deep inside: Autoencoders,” Medium, (Apr. 10, 2018), [Online]. Available: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f (visited on 10/16/2024).

[2] “Introduction to autoencoders.,” Jeremy Jordan, (Mar. 19, 2018), [Online]. Available: https://www.jeremyjordan.me/autoencoders/ (visited on 10/16/2024).

[3] “Autoencoders — deep learning bits #1 | HackerNoon,” (), [Online]. Available: https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694 (visited on 10/16/2024).

Bibliography I

[1] N. Hubens, “Deep inside: Autoencoders,” Medium, (Apr. 10, 2018), [Online]. Available: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f (visited on 10/16/2024).

[2] “Introduction to autoencoders.,” Jeremy Jordan, (Mar. 19, 2018), [Online]. Available: https://www.jeremyjordan.me/autoencoders/ (visited on 10/16/2024).

[3] “Autoencoders — deep learning bits #1 | HackerNoon,” (), [Online]. Available: https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694 (visited on 10/16/2024).