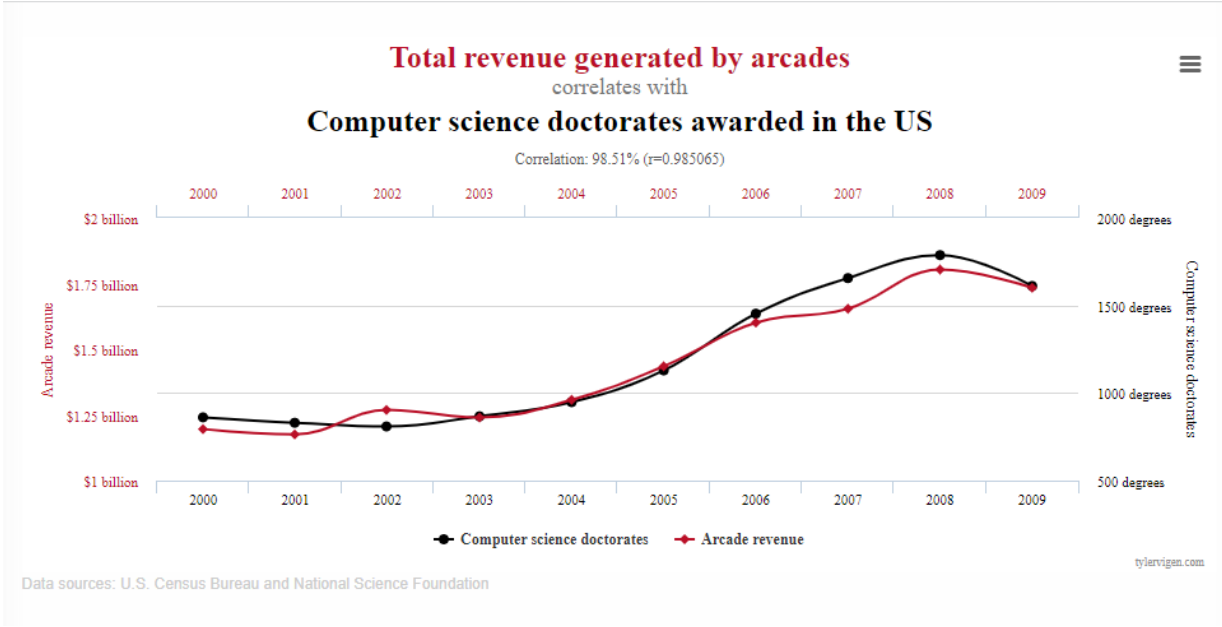
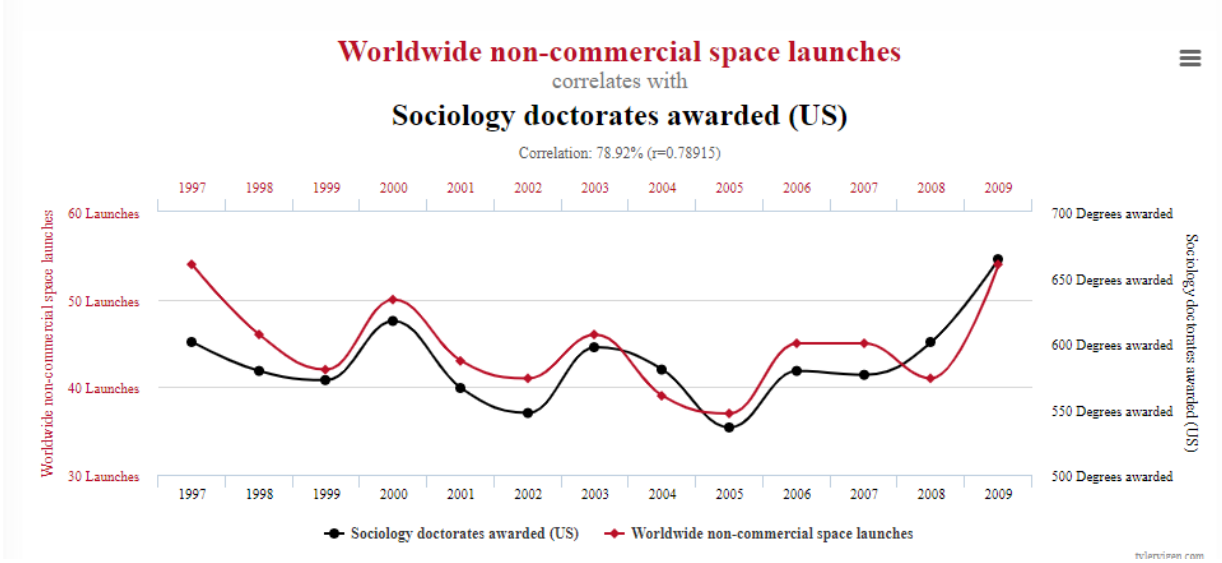
THE UNIVERSITY of TENNESSEE KNOXVILLE

Day 5: Deep Learning

Sergei V. Kalinin

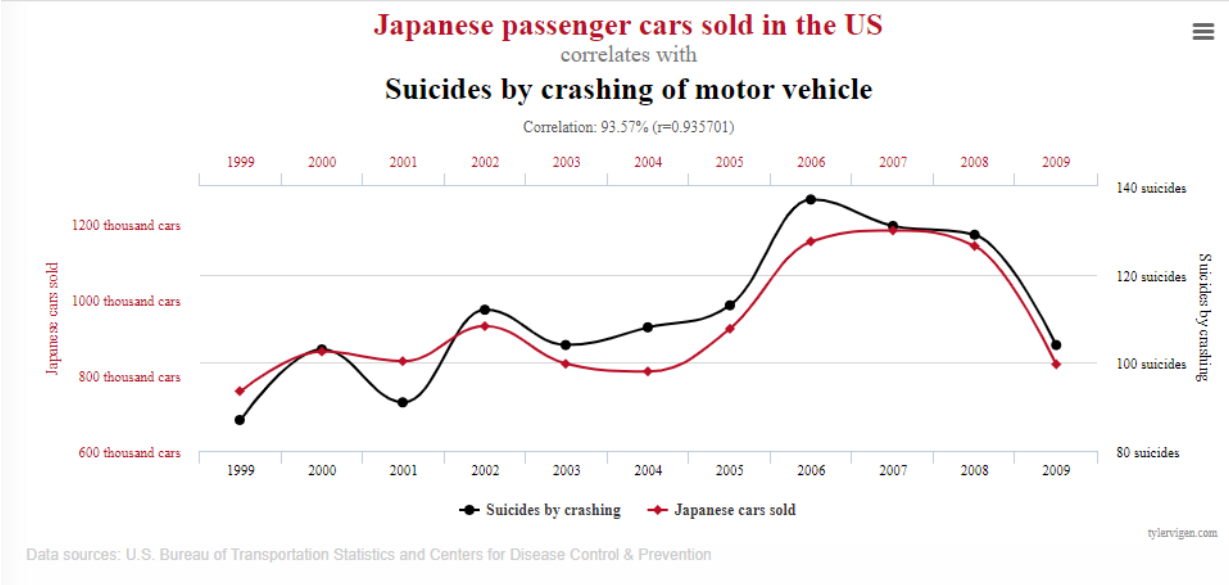
Correlation and causation

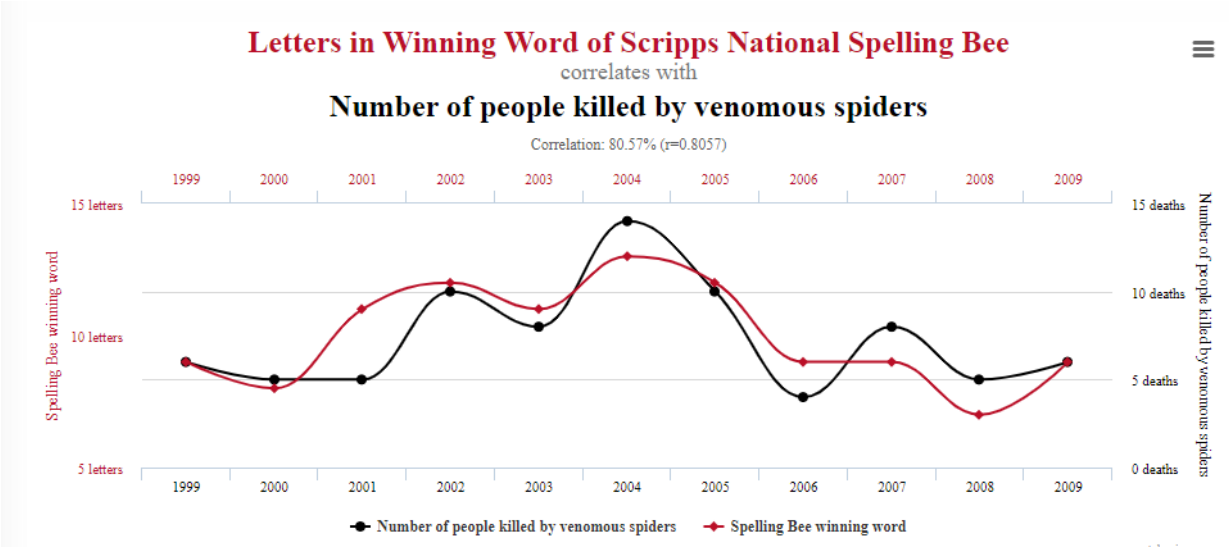




https://www.tylervigen.com/spurious-correlations

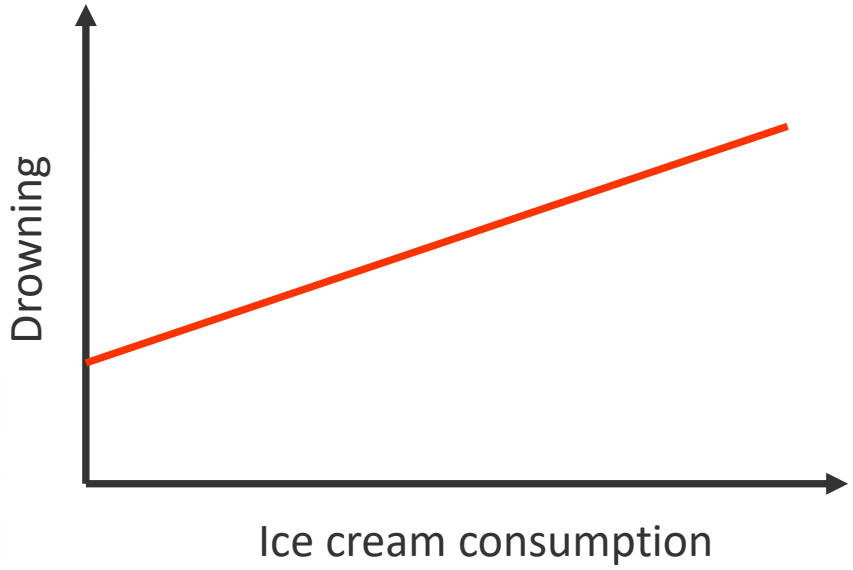
Correlation and causation

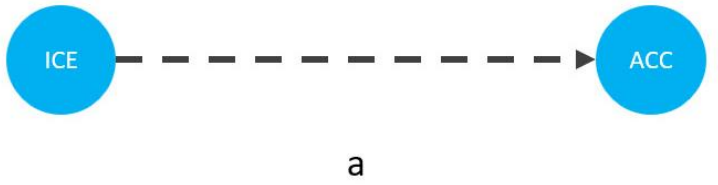




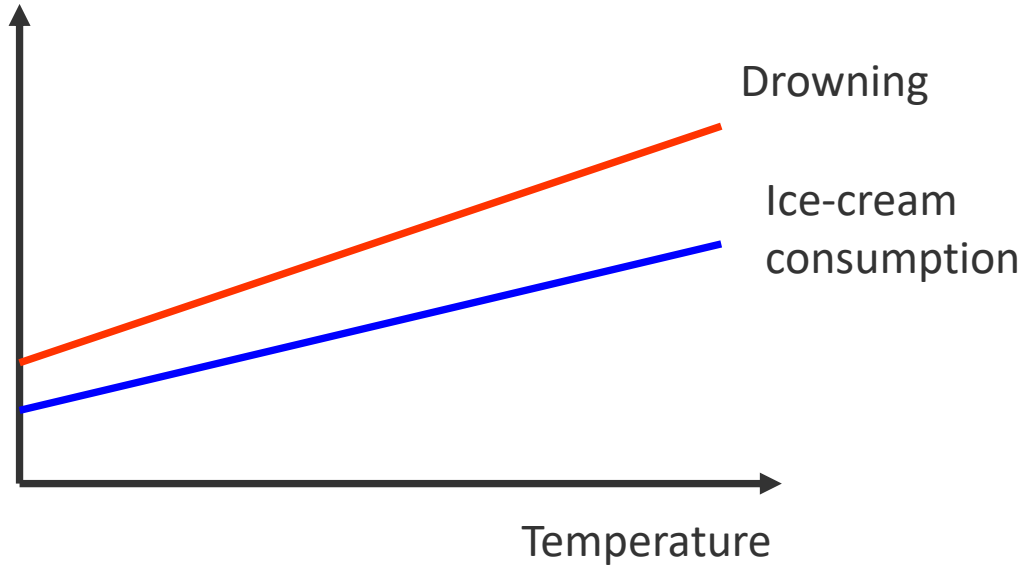
https://www.tylervigen.com/spurious-correlations

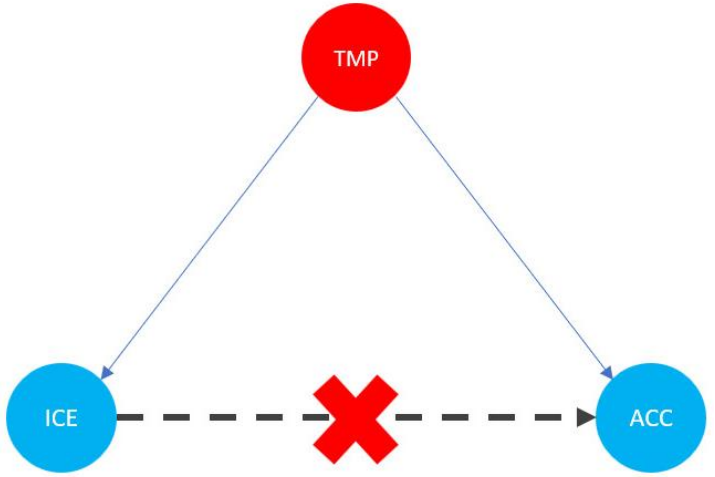
Ice-cream and drowning





a

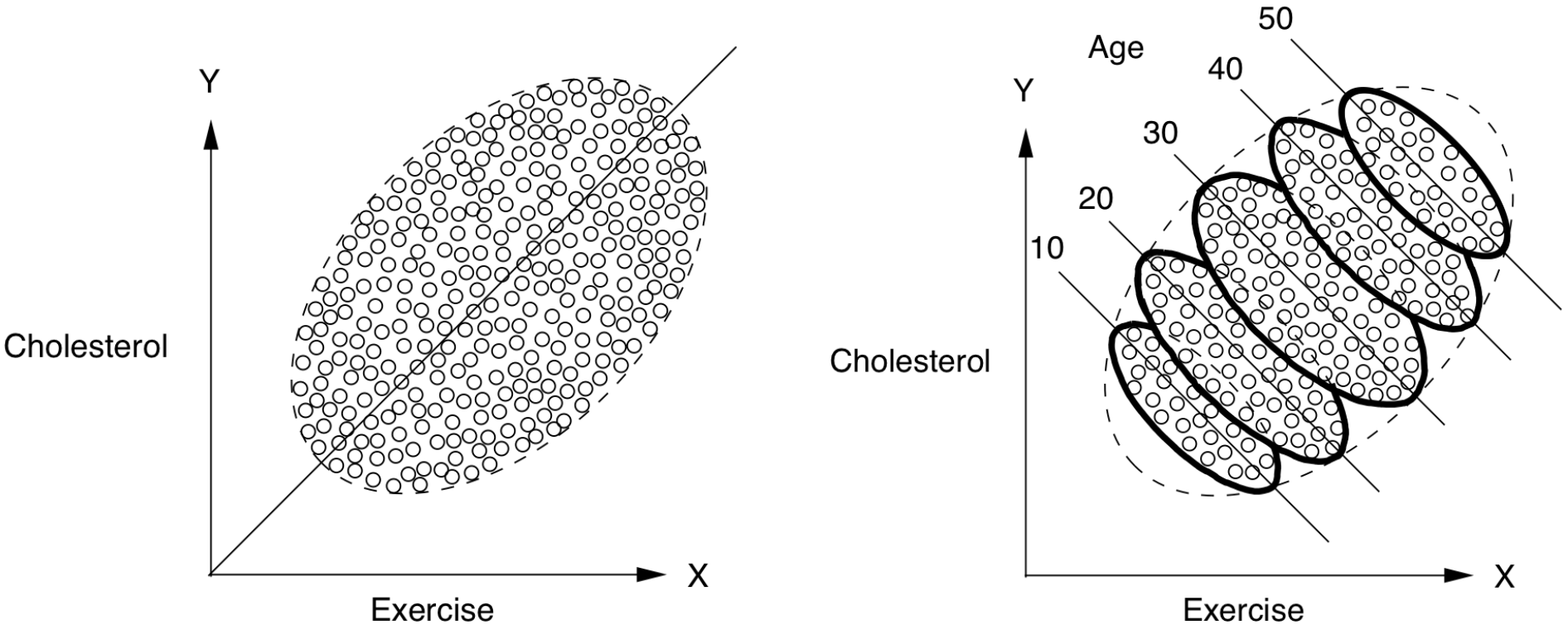




b

From A. Molak, Causal Inference and Discovery in Python

Simpson paradox



Exercise is helpful in every age group but harmful for a typical person.

Is exercise helpful or not?

From J. Pearl presentation, Why - 19

Berkeley discrimination lawsuit

In the early 1970s, the University of California, Berkeley was sued for gender discrimination over admission to graduate school. Of the 8,442 male applicants for the fall of 1973, 44 percent were admitted, but only 35 percent of the 4,351 female applicants were accepted

Table 1: Data From Six Largest Departments of 1973 Berkeley Discrimination Case

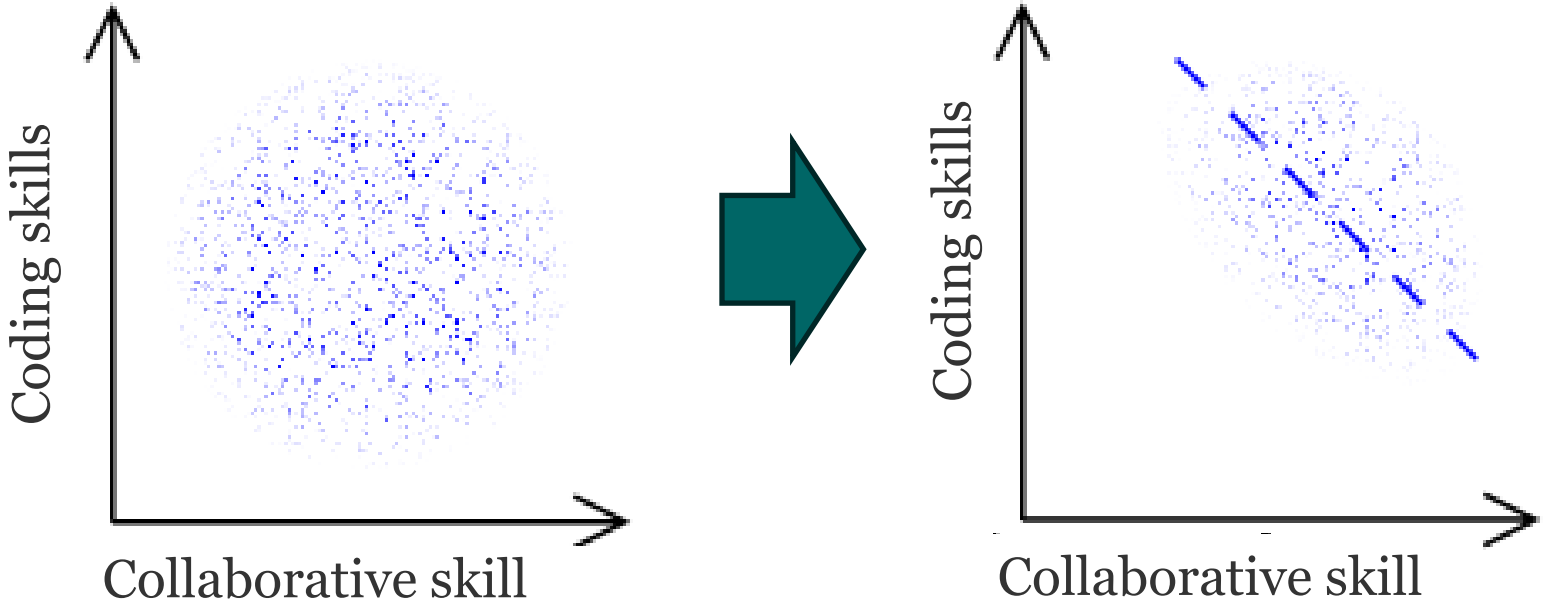
Department	Men		Women	
	Applicants	Admitted	Applicants	Admitted
A	825	62%	108	82%
B	560	63%	25	68%
C	325	37%	593	34%
D	417	33%	375	35%
E	191	28%	393	24%
F	272	6%	341	7%

Source: Bickel, Hammel, and O'Connell (1975); table accessed via Wikipedia at https://en.wikipedia.org/wiki/Simpson%27s_paradox

In the Berkeley case, the "paradox" occurred because women disproportionately applied to departments with low acceptance rates, as shown in the table above, while men disproportionately applied to departments with high acceptance rates.

https://www.brookings.edu/articles/when-average-isnt-good-enough-simpsons-paradox-in-education-and-earnings/

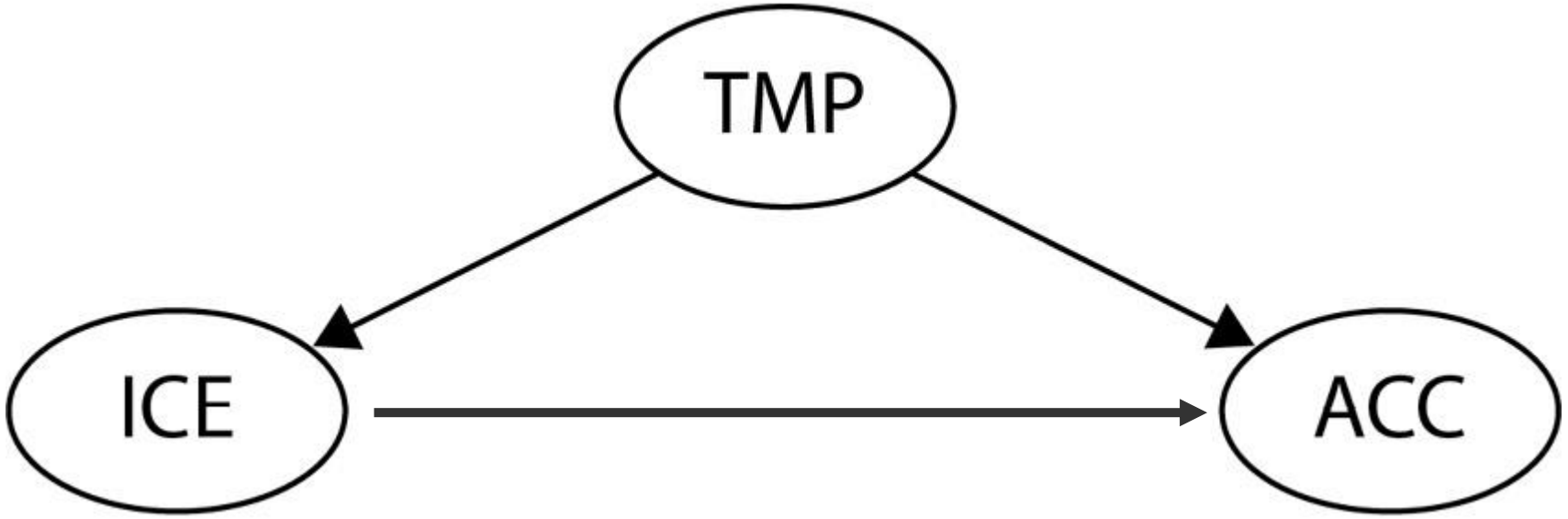
Colliders and Berkson paradox



Many companies might hire people based on their skills and their personality traits. Imagine that company *X* quantifies a person's coding skills on a scale from one to five. They do the same for the candidate's ability to cooperate and hire everyone who gets a total score of at least seven. Assuming that coding skills and ability to cooperate are independent in the population (which doesn't have to be true in reality), you'll observe that in company *X*, people who are better coders are less likely to cooperate on average, and those who are more likely to cooperate have fewer coding skills. You could conclude that being non-cooperative is related to being a better coder, yet this conclusion would be incorrect in the general population.

https://en.wikipedia.org/wiki/Berkson%27s_paradox From A. Molak, Causal Inference and Discovery in Python

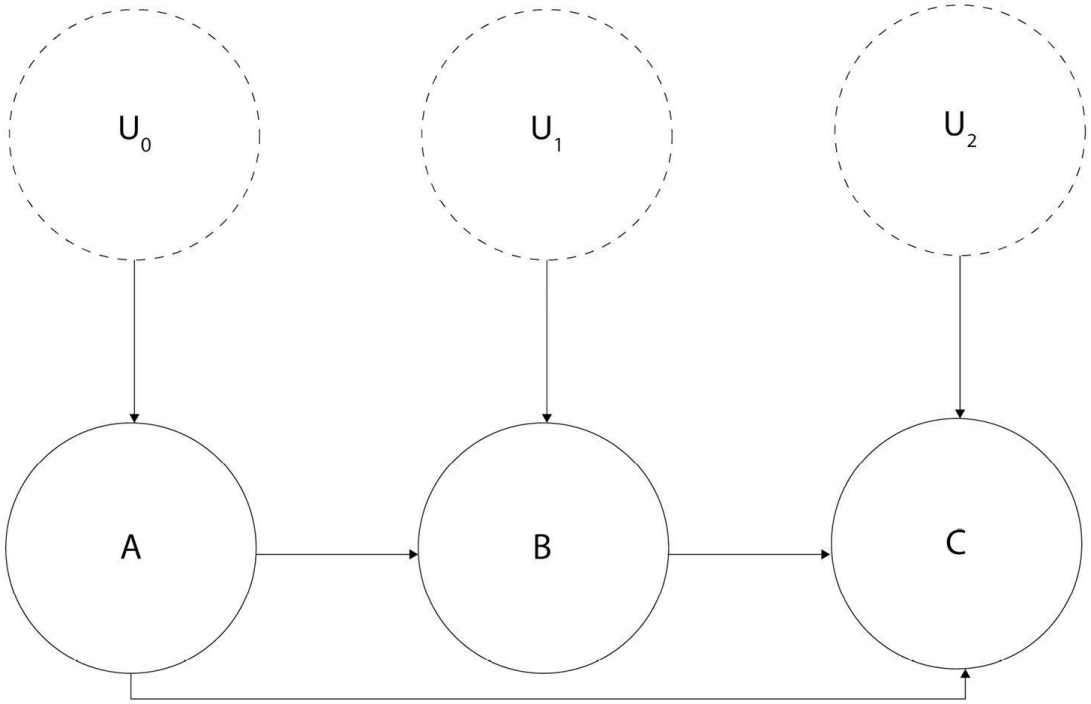
How can we even approach such problems?



- Observations give us correlations between temperature, ice cream consumption, and accident rate
- What we need to know is the causal links between these characteristics. Does change in ice cream consumption affect temperature or accident rate?
- But we cannot make an experiment!

From A. Molak, Causal Inference and Discovery in Python

Causal graphs



$$A := f_A(U_0)$$

$$B := f_B(A, U_1)$$

$$C := f_C(A, B, U_2)$$

- Here, := is an **assignment operator**, also known as a **walrus operator**. We use it to emphasize that the relationship that we're describing is *directional* (or asymmetric), as opposed to the regular equal sign that suggests a symmetric relation.

- And f_A, f_B, f_C represent arbitrary functions (they can be as simple as a summation or as complex as you want).

From A. Molak, Causal Inference and Discovery in Python

Do-operator

Conditioning: $P(X = x | Y = y)$

Intervention: $P(Y = 1 | do(X = 0))$

- Conditioning only modifies our *view* of the data, while intervening affects the distribution by *actively* setting one (or more) variable(s) to a *fixed value* (or a distribution).
- This is very important – intervention *changes* the system, but conditioning *does not*.
- You might ask, what does it mean that *intervention changes the system*? Great question!

From A. Molak, Causal Inference and Discovery in Python

How can we learn causality

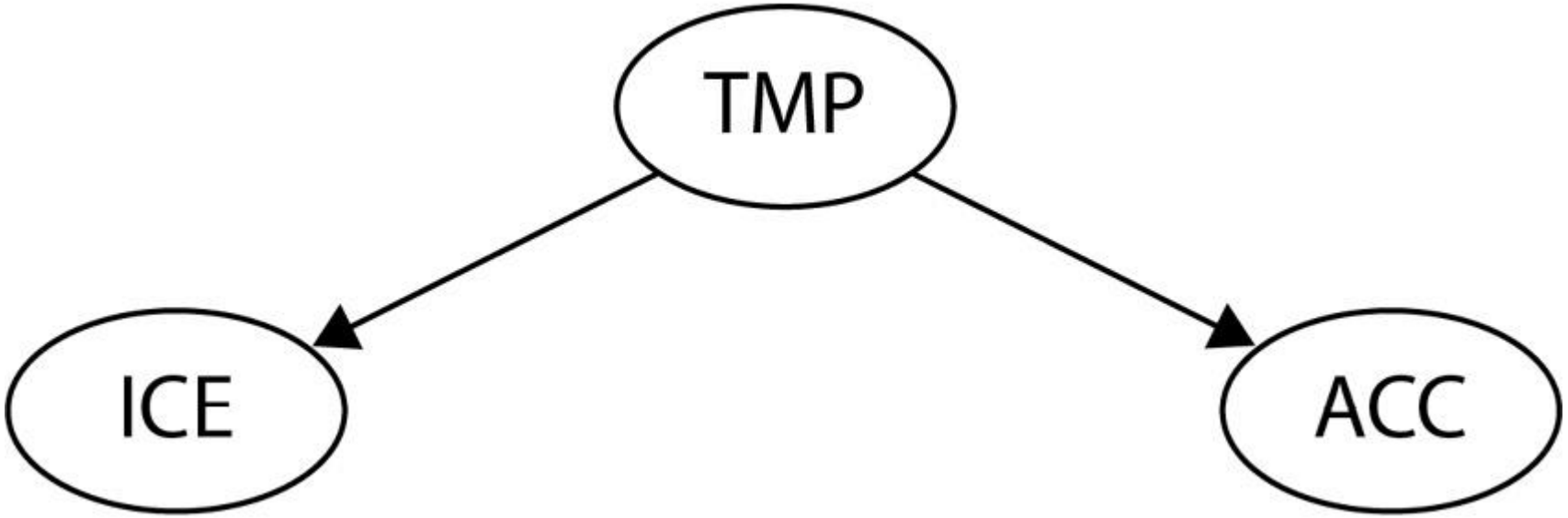
- **Causal discovery** and **causal structure learning** are umbrella terms for various kinds of methods used to uncover causal structure from observational or interventional data.

- **Expert knowledge** is a term covering various types of knowledge that can help define or disambiguate causal relations between two or more variables. Depending on the context, expert knowledge might refer to knowledge from randomized controlled trials, laws of physics, a broad scope of experiences in a given area, and more.

- **Combining causal discovery and expert knowledge:** Some causal discovery algorithms allow us to easily incorporate expert knowledge as a priority. This means that we can either *freeze* certain edges in the graph or *suggest* the existence or direction of these edges.

From A. Molak, Causal Inference and Discovery in Python

Adjustment

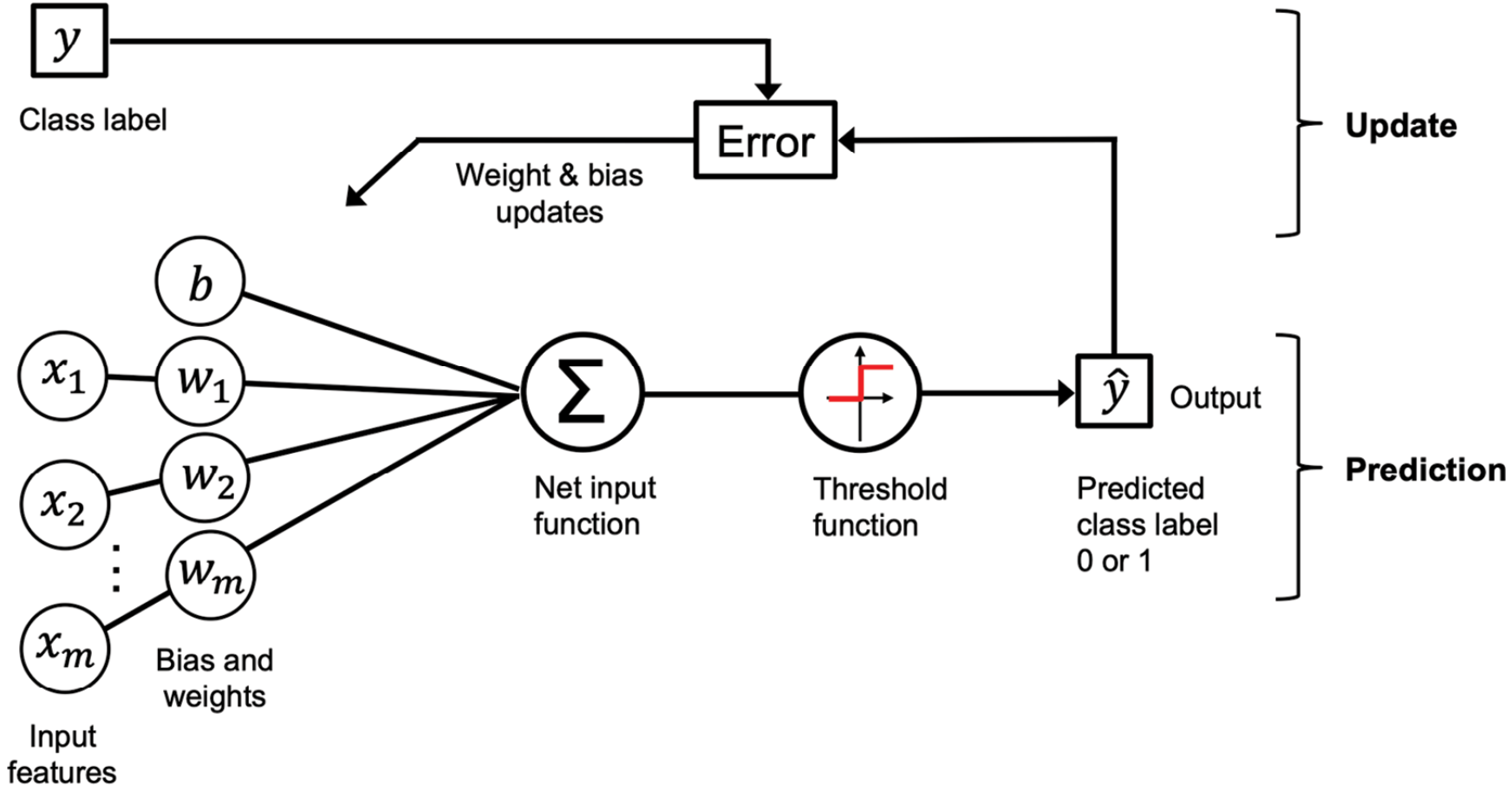


$$ACC \sim ICE + TMP$$

$$P(ACC|do(ICE)) = \sum_{tmp} P(ACC|ICE, TMP)P(TMP)$$

From A. Molak, Causal Inference and Discovery in Python

Training Linear Neuron



From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Training Linear Neuron

- Initialize the weights and bias unit to 0 or small random numbers
- For each training example, ***x(i)***:
- Compute the output value, $\boldsymbol{y(i) = w^T x(i) + b}$
- Update the weights and bias unit: $w_j := w_j + \Delta w_j$ and $b := b + \Delta b$
- Where $\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$ and $\Delta b = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)$

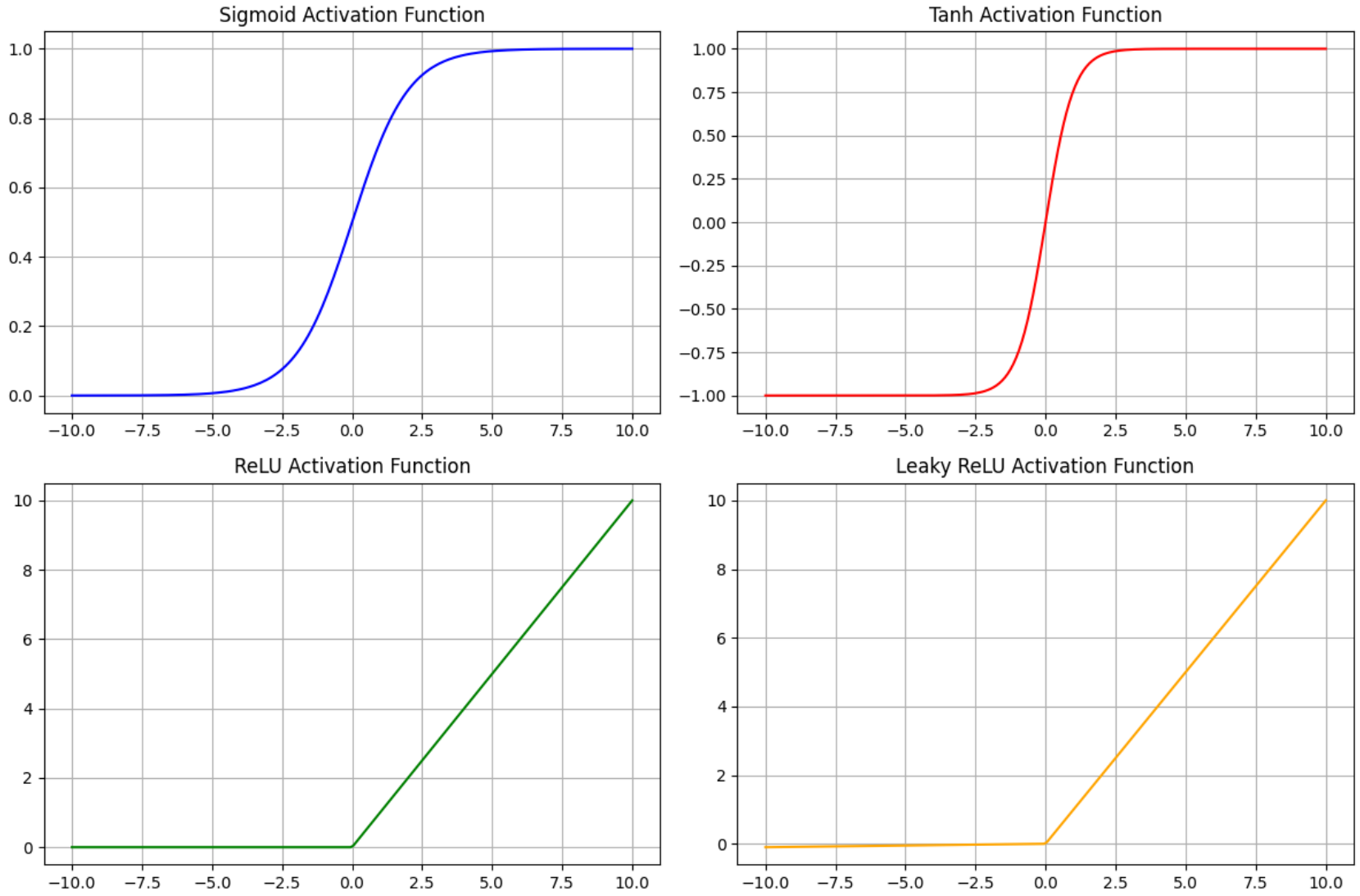
Each weight, w_j, corresponds to a feature, x_j, in the dataset,

η is the **learning rate** (typically a constant between 0.0 and 1.0),

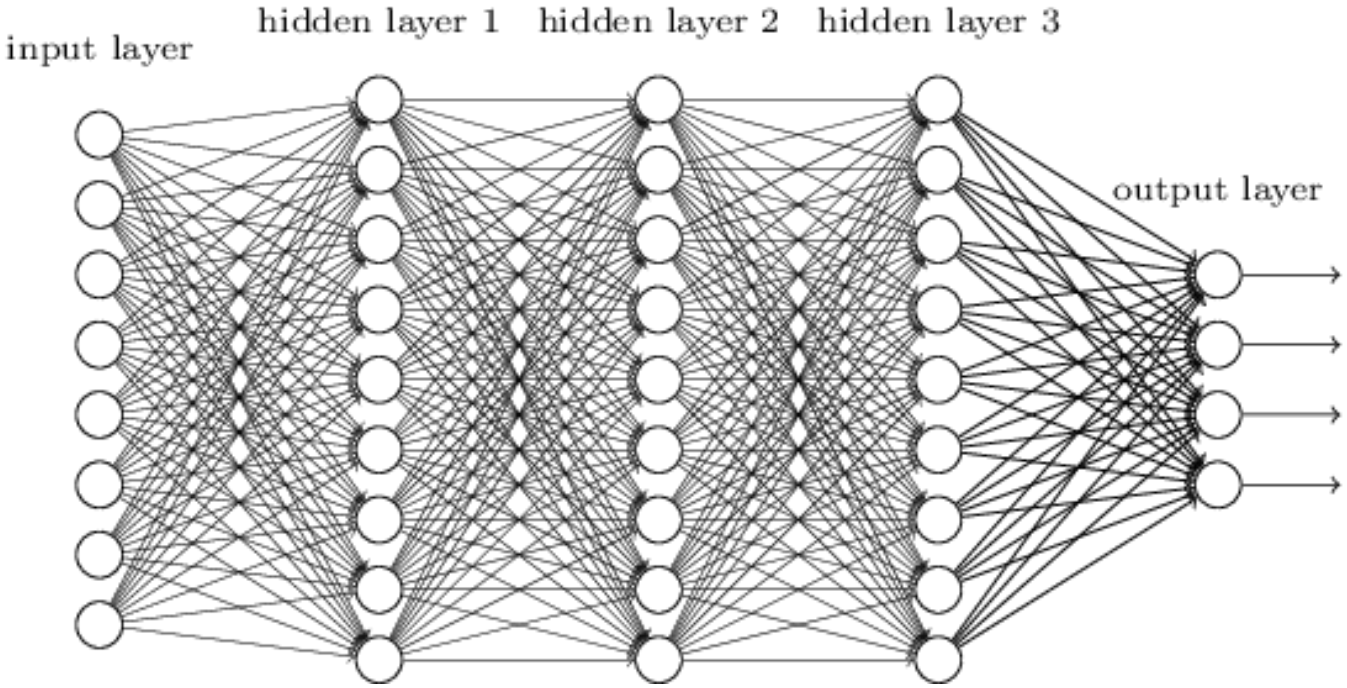
$y^{(i)}$ is the **true class label** of the *i*th training example,

$\hat{y}^{(i)}$ is the **predicted class label**

Activation functions



Putting Neurons Together



- Composed of multiple layers of artificial neurons.
- Each layer processes inputs received, applies a transformation (weights, biases, activation function), and passes the output to the next layer.
- Training a DNN involves adjusting weights and biases using backpropagation and a chosen optimization algorithm.
- The deep architecture enable the network to learn complex and abstract patterns in data.

Loss functions for supervised ML

A loss function, also known as a cost function, quantifies the difference between the predicted values and the actual target values. It guides the training of neural networks by providing a measure to minimize during optimization

Mean Squared Error (MSE): Used for regression problems.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Measures the average squared difference between actual and predicted values

Cross-Entropy Loss: Used for classification problems.

$$CE = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

Measures the performance of a classification model whose output is a probability value between 0 and 1

- Loss functions provide the primary feedback signal for learning.
- The choice of loss function can significantly affect the model's performance and convergence

Backpropagation

Backpropagation is a mechanism used to update the weights in a neural network efficiently, based on the error rate obtained in the previous epoch (i.e., iteration). It effectively distributes the error back through the network layers

- **Forward Pass:** Calculating the predicted output, moving the input data through the network layers
- **Loss Function:** Determining the error by comparing the predicted output to the actual output
- **Backward Pass:** Computing the gradient of the loss function with respect to each weight by the chain rule
- **Weight Update:** Adjusting the weights of the network in a direction that minimally reduces the loss (gradient descent)

Input Data → Forward Pass → Calculate Loss → Backward Pass → Update Weights

https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

Metrics

Quantitative measures used to evaluate and monitor the performance of the network, both during training and after deployment. Here are some common metrics used in DCNNs:

Accuracy: Defined as the ratio of correctly predicted observations to the total observations.

Precision: The ratio of correctly predicted positive observations to the total predicted positive observations. Important in scenarios where the cost of a false positive is high.

Recall (Sensitivity): The ratio of correctly predicted positive observations to all observations in the actual class. Crucial in situations where missing a positive instance is costly.

F1 Score: The harmonic mean of precision and recall. Trade-off between precision and recall.

Mean Squared Error (MSE): Commonly used in regression tasks.

Area Under the ROC Curve (AUC-ROC): Represents the degree or measure of separability achieved by the model.

Intersection over Union (IoU): Typically used in object detection and segmentation tasks. Measures the overlap between the predicted bounding box and the ground truth box.

How is metrics different from loss function?

- **Optimization vs. Evaluation:** Loss functions are used to optimize the model (i.e., to adjust the model parameters during training), whereas metrics are used to evaluate the model's performance.
- **Differentiability:** Loss functions need to be differentiable with respect to model parameters, but metrics do not have this requirement.
- **Interpretability:** Metrics are often more interpretable than loss functions and are chosen based on what makes the most sense for the specific application or domain.
- **Role in Model Training:** The loss function directly influences the training process, while metrics are more about assessing the outcome of that training.

How to not get lost?

- A vast array of network architectures ranging from Multilayer Perceptrons (MLPs) to Graph Neural Networks and Transformers
- Each architecture has unique characteristics suited for different types of data and tasks, ways to define the architecture, and so on
- Numerous methods to engineer loss functions
- Numerous ways to implement regularization

Problem-Centric Approach:

- Always start with the problem at hand: Analyze the nature of input data and desired output
- Choose a network architecture that aligns with the type and structure of your data
- Select a loss function that reflects the objective of the problem
- Metrics should be chosen based on what measures success for your specific task

Hyperparameter Tuning:

- Once the architecture and loss function are set, proceed to tune hyperparameters including network structure, optimizers, regularization, etc.
- Hyperparameter tuning should be guided by the chosen metrics and the nature of the problem

Some useful resources:

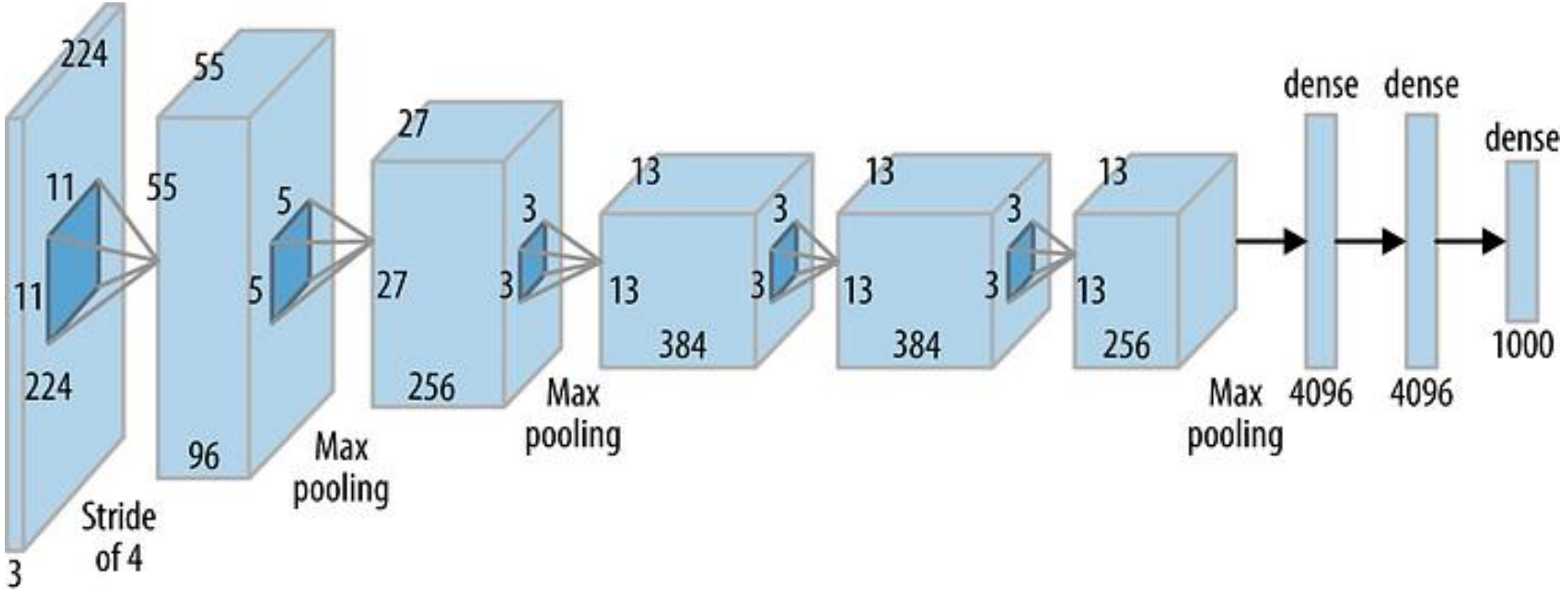
https://udlbook.github.io/udlbook/

https://dmol.pub/ml/classification.html

https://keras.io/examples/

Deep Convolutional Neural Networks

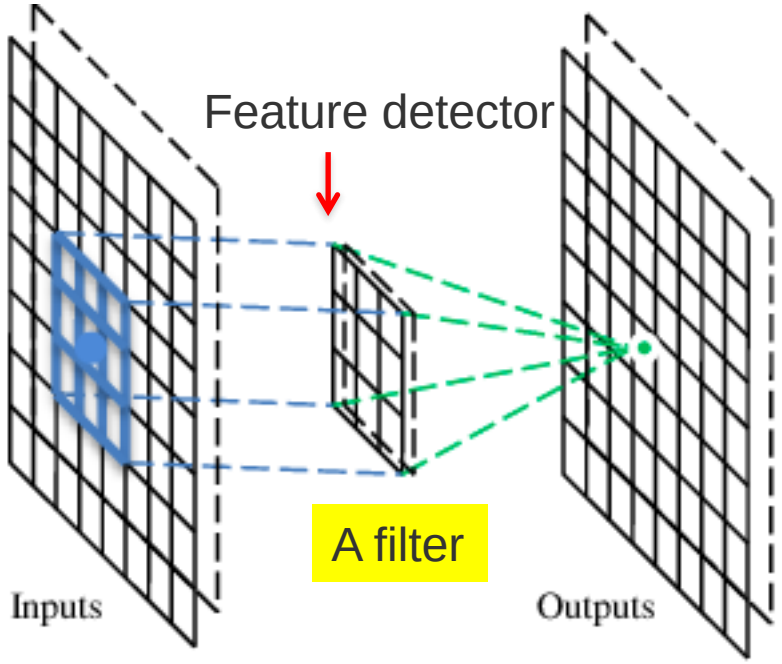
Structure of AlexNet



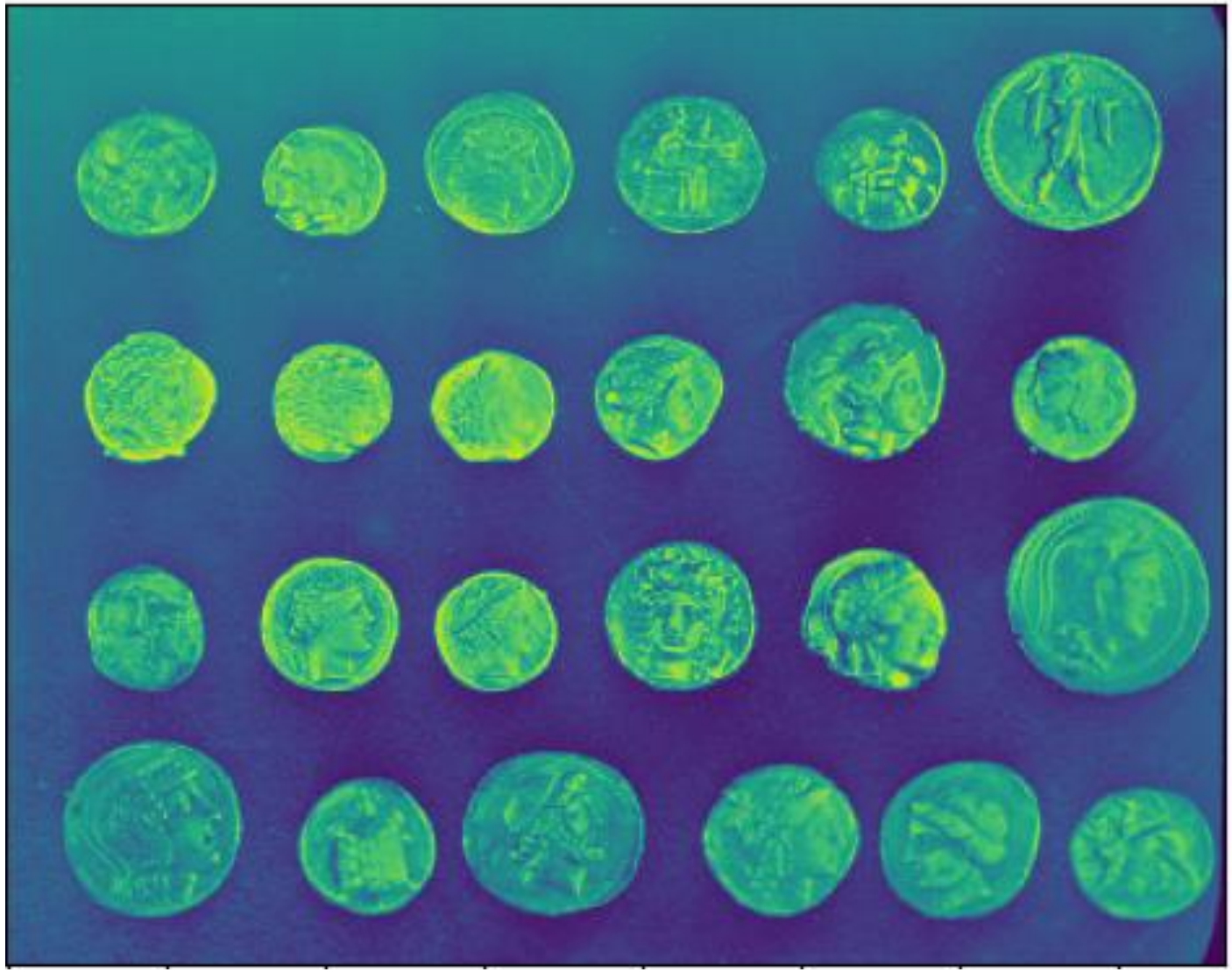
https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

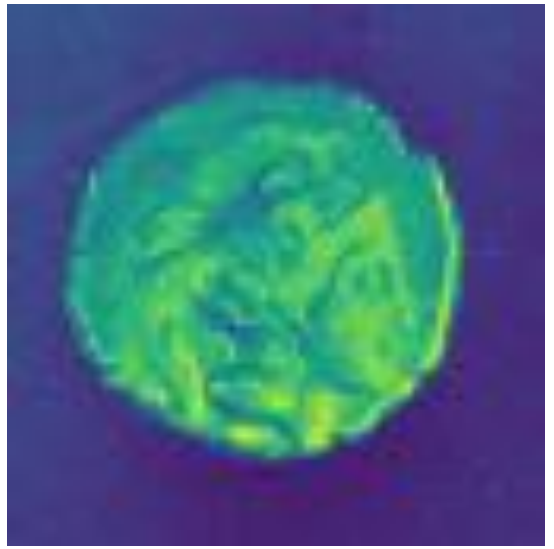
Finding features in images

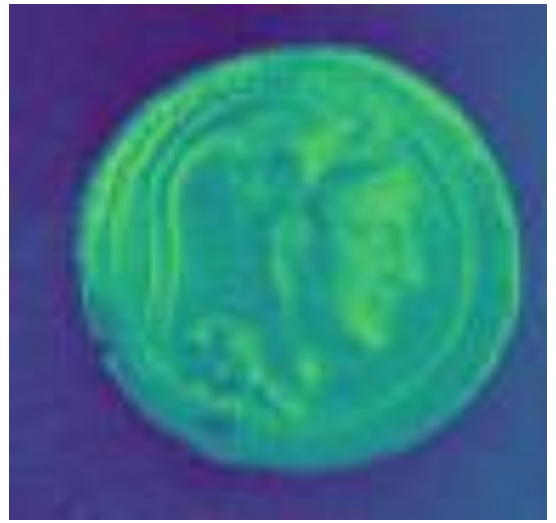
A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Coin example (scikit-image)

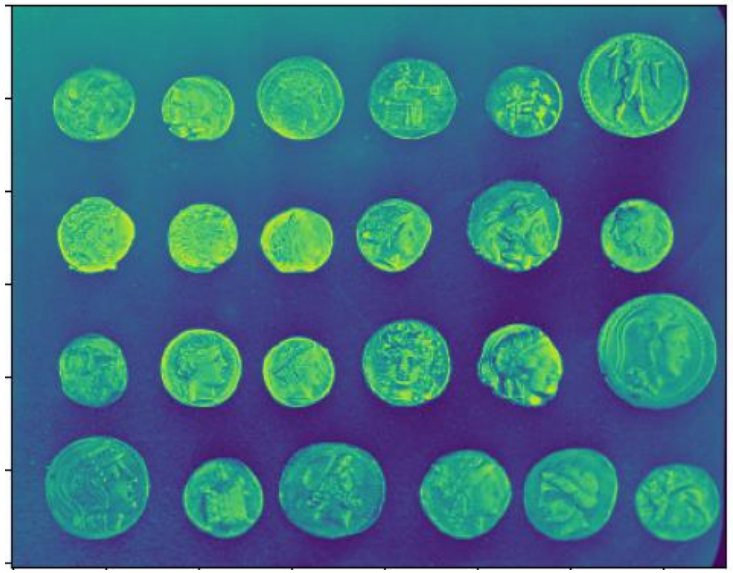


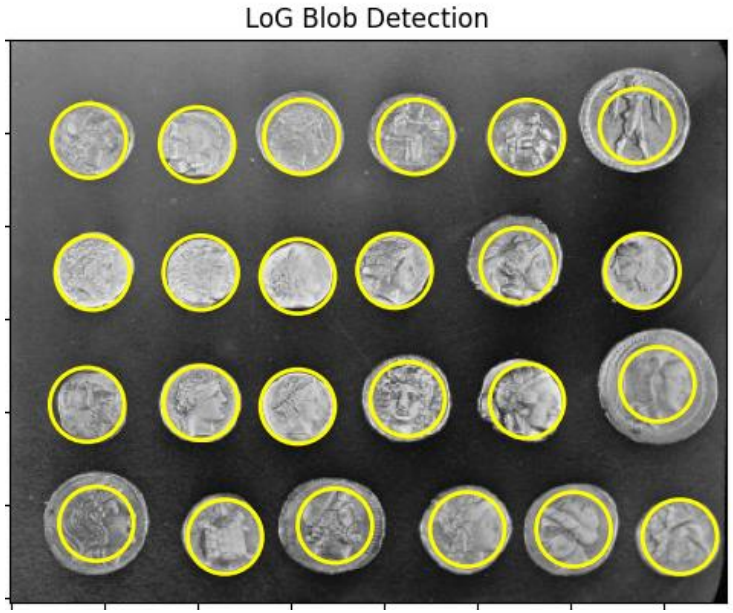


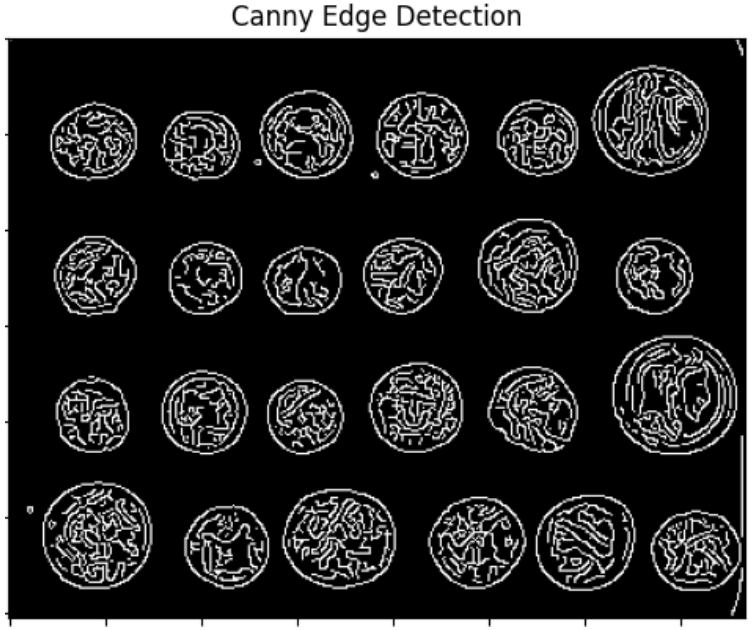


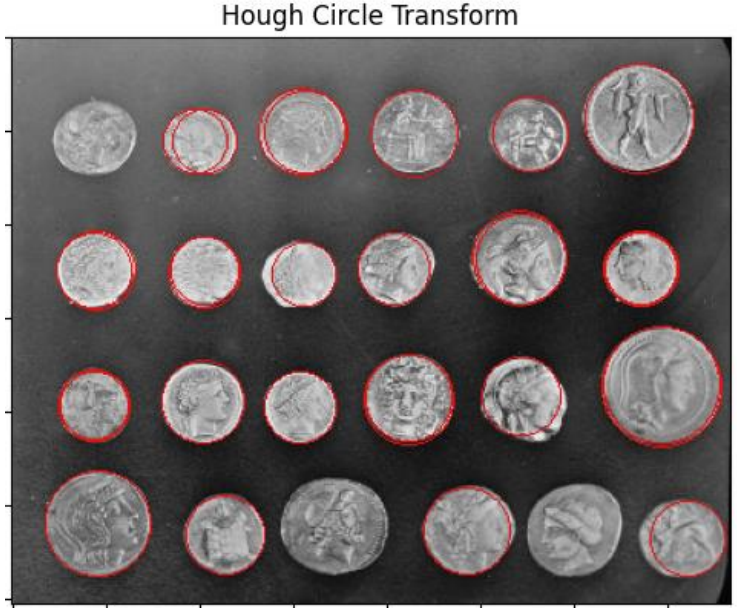
What are features in this image?

Feature finding in scikit-image

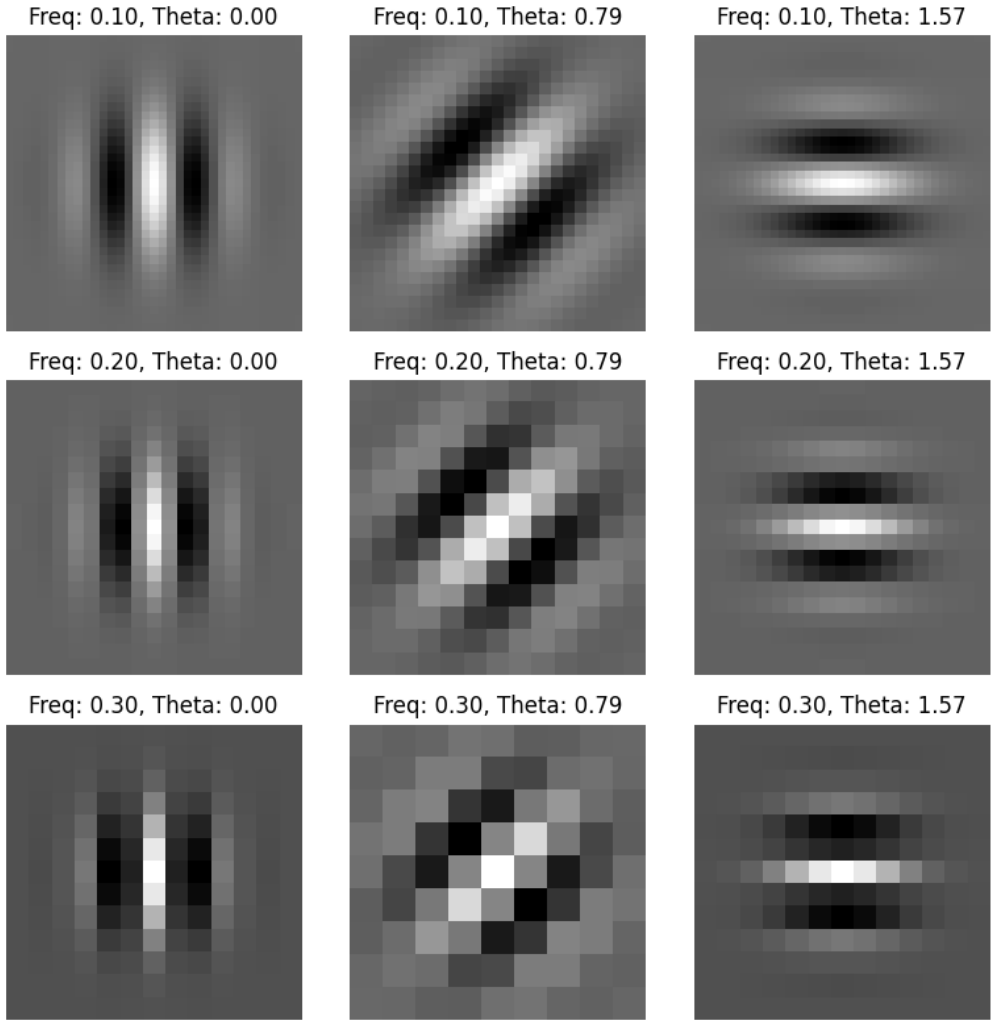


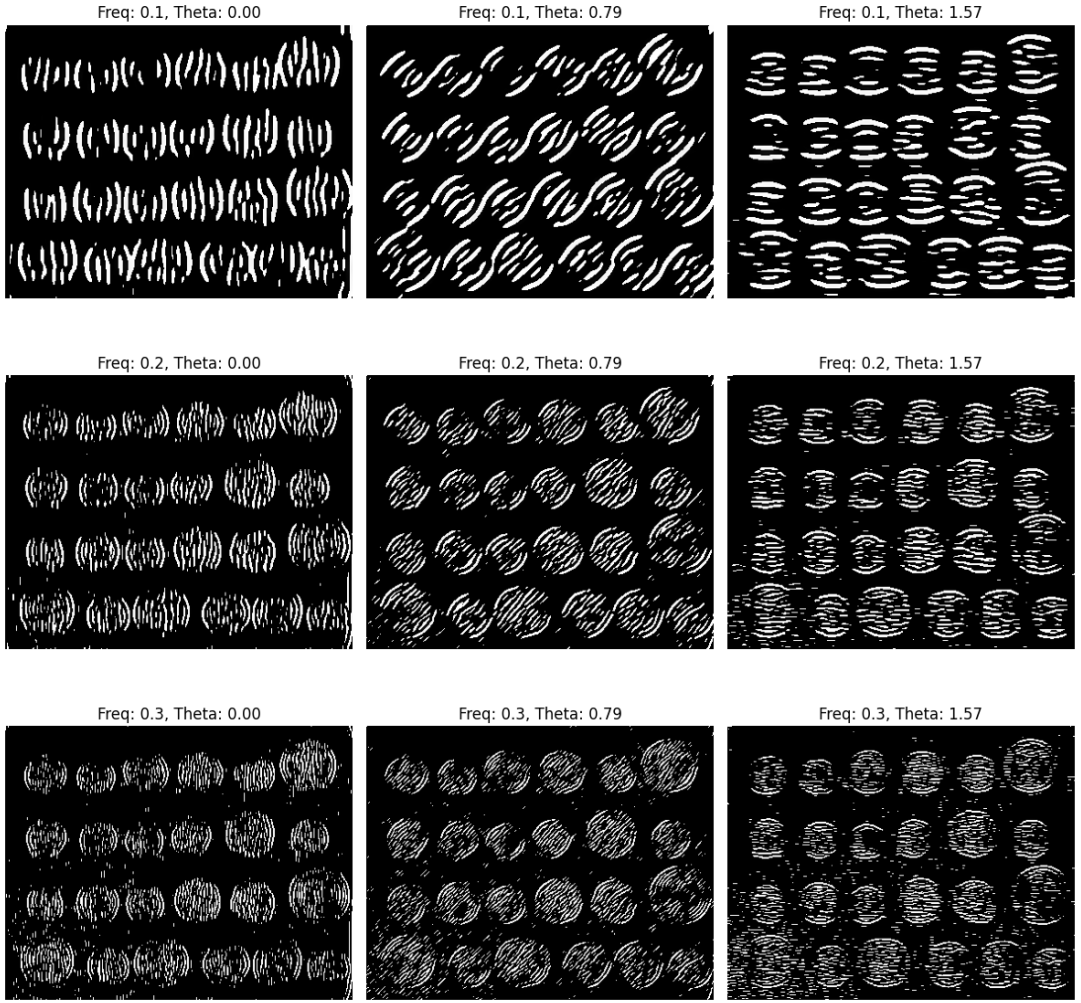






Gabor filters





What is convolution?

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

These are the network parameters to be learned.

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).

What is convolution?

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

Filter 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

Dot product → 3 -1

6 x 6 image

What is convolution?

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

Filter 1

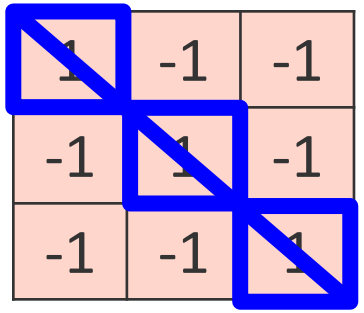
If stride=2

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

6 x 6 image

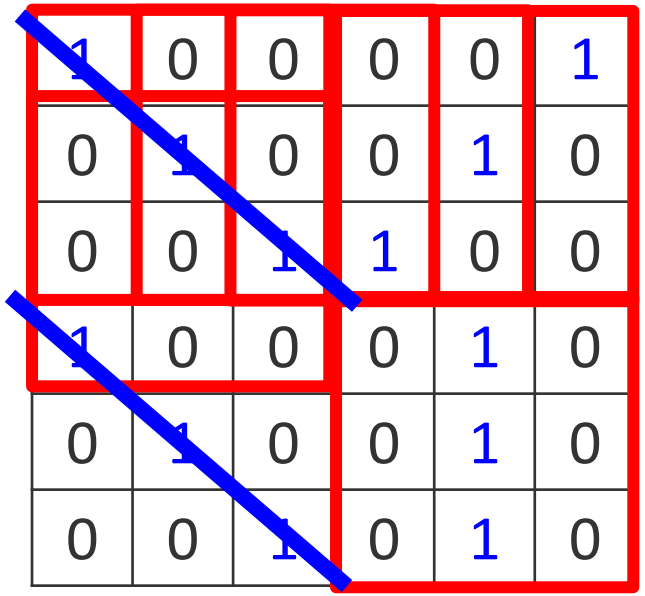
3 -3

What is convolution?

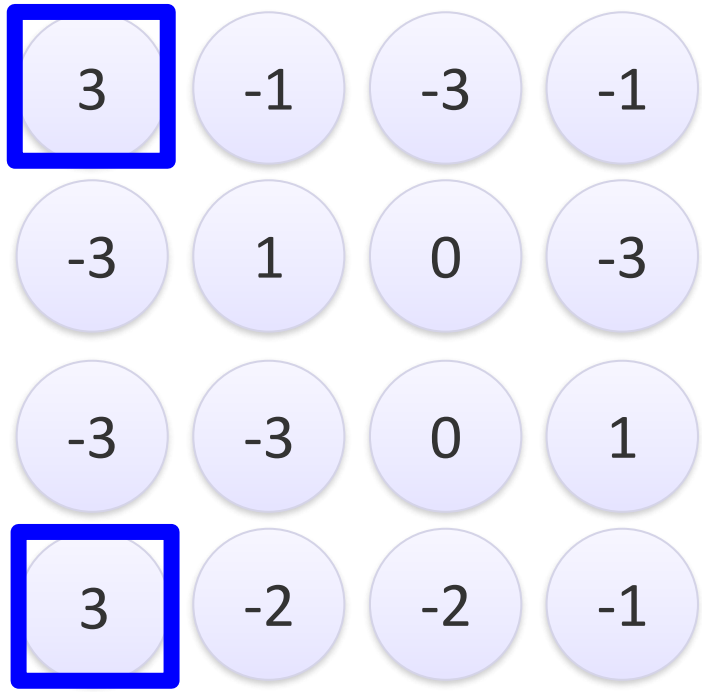


Filter 1

stride=1



6 x 6 image



What is convolution?

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

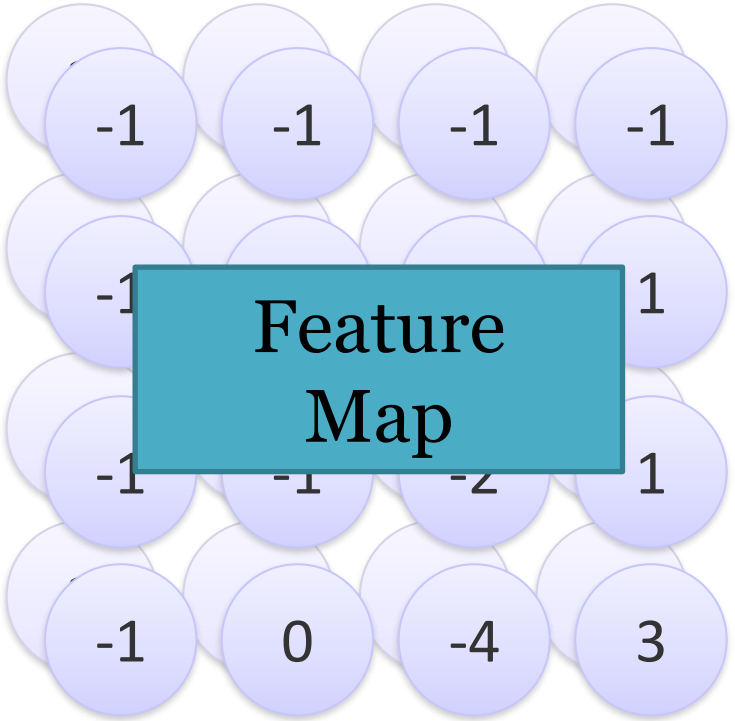
Filter 2

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

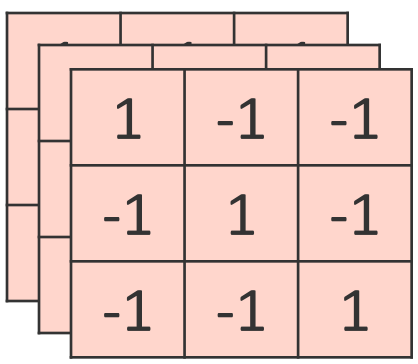
6 x 6 image

Repeat this for each filter

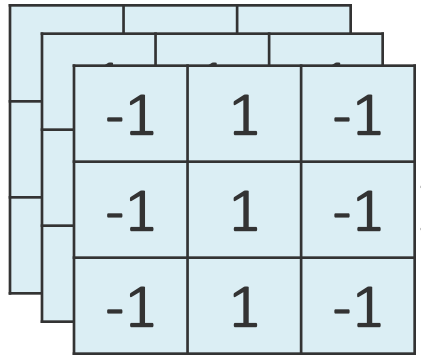


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

But what about color?

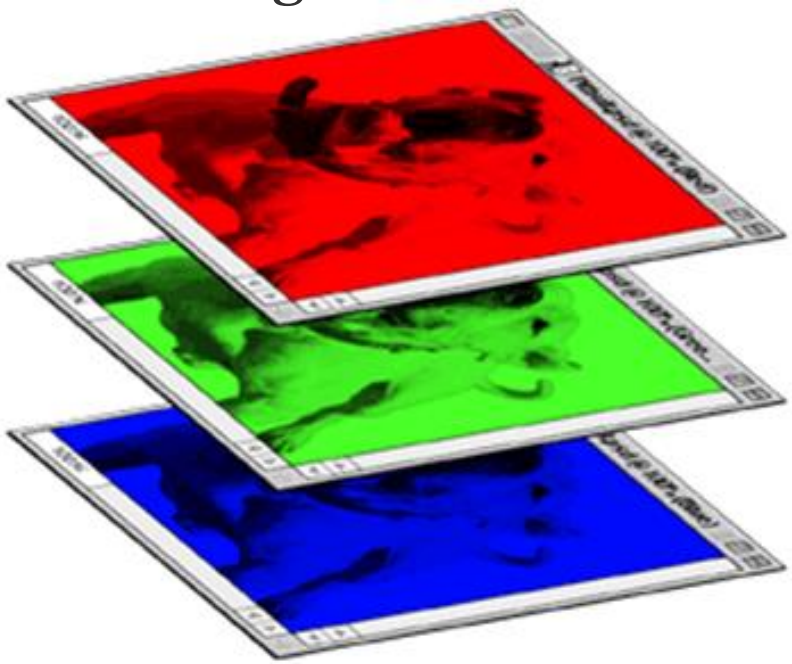


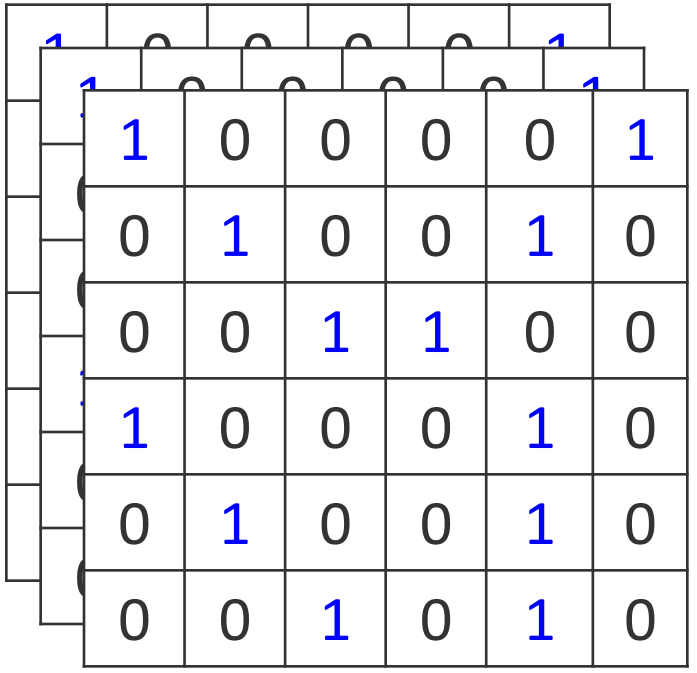
Filter 1



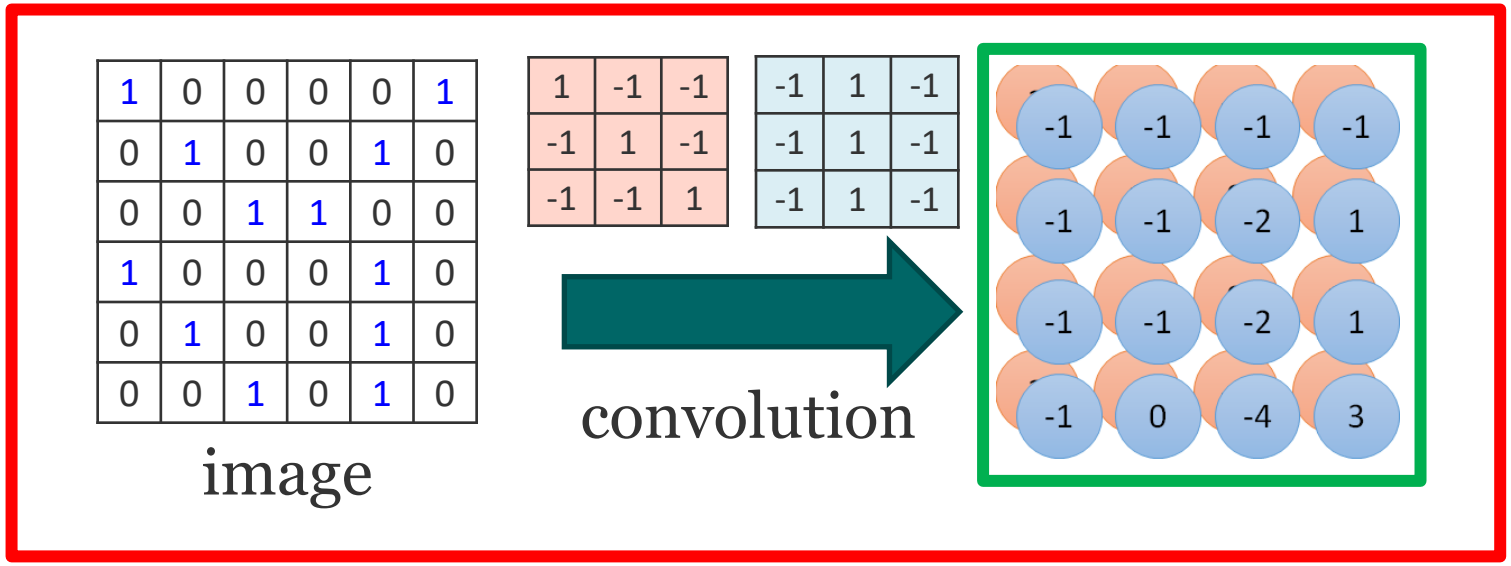
Filter 2

Color image

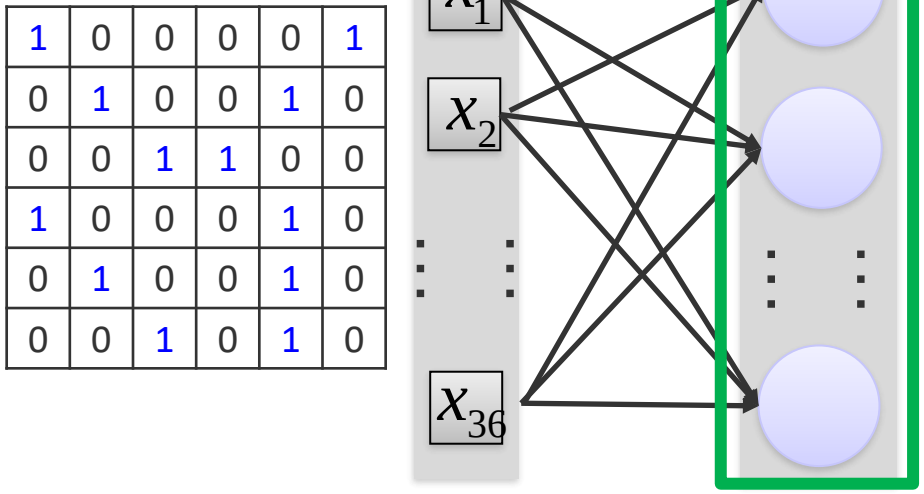


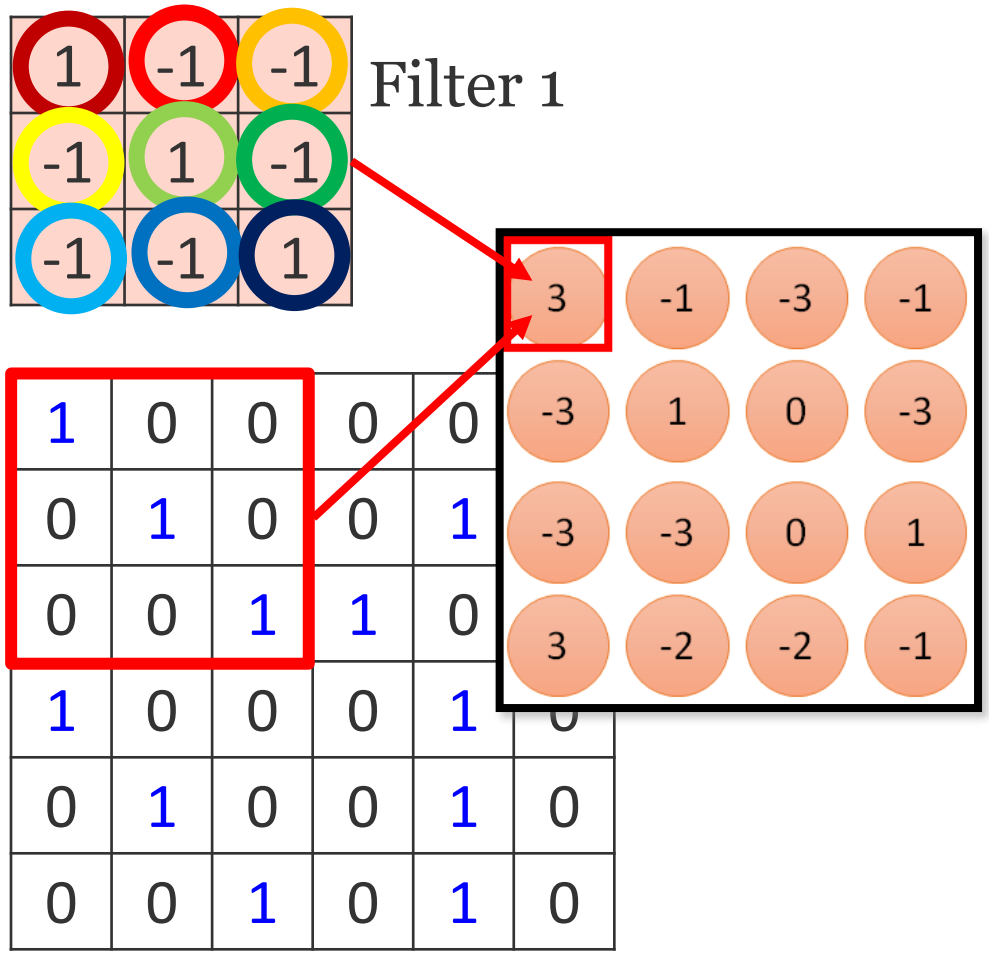


Convolution vs. fully connected

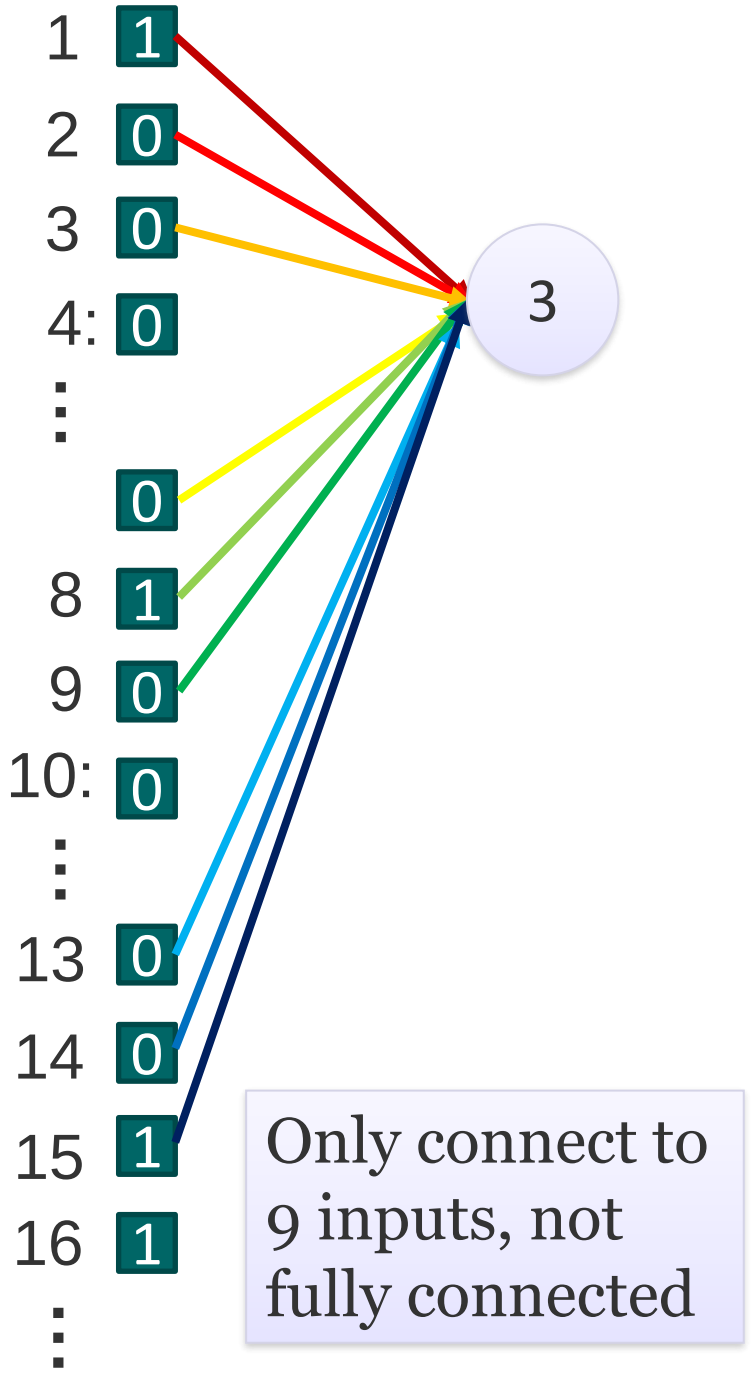


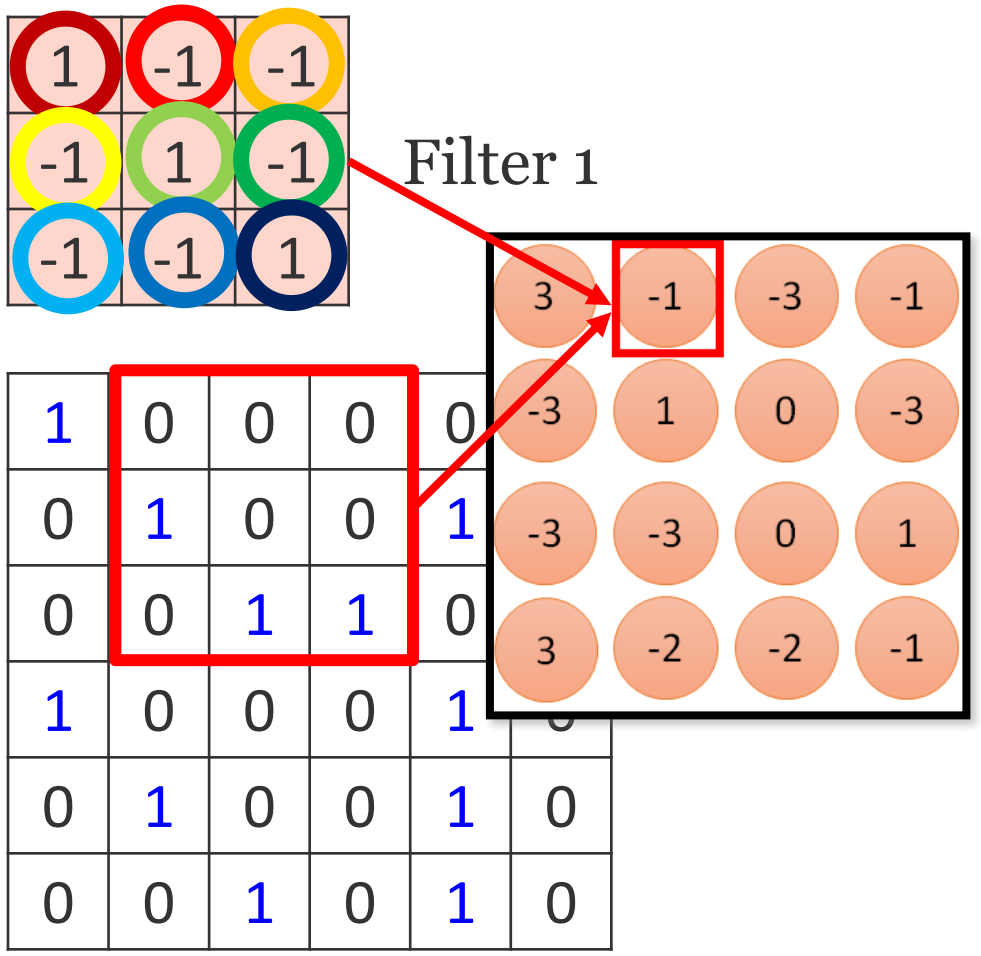
Fully-connected



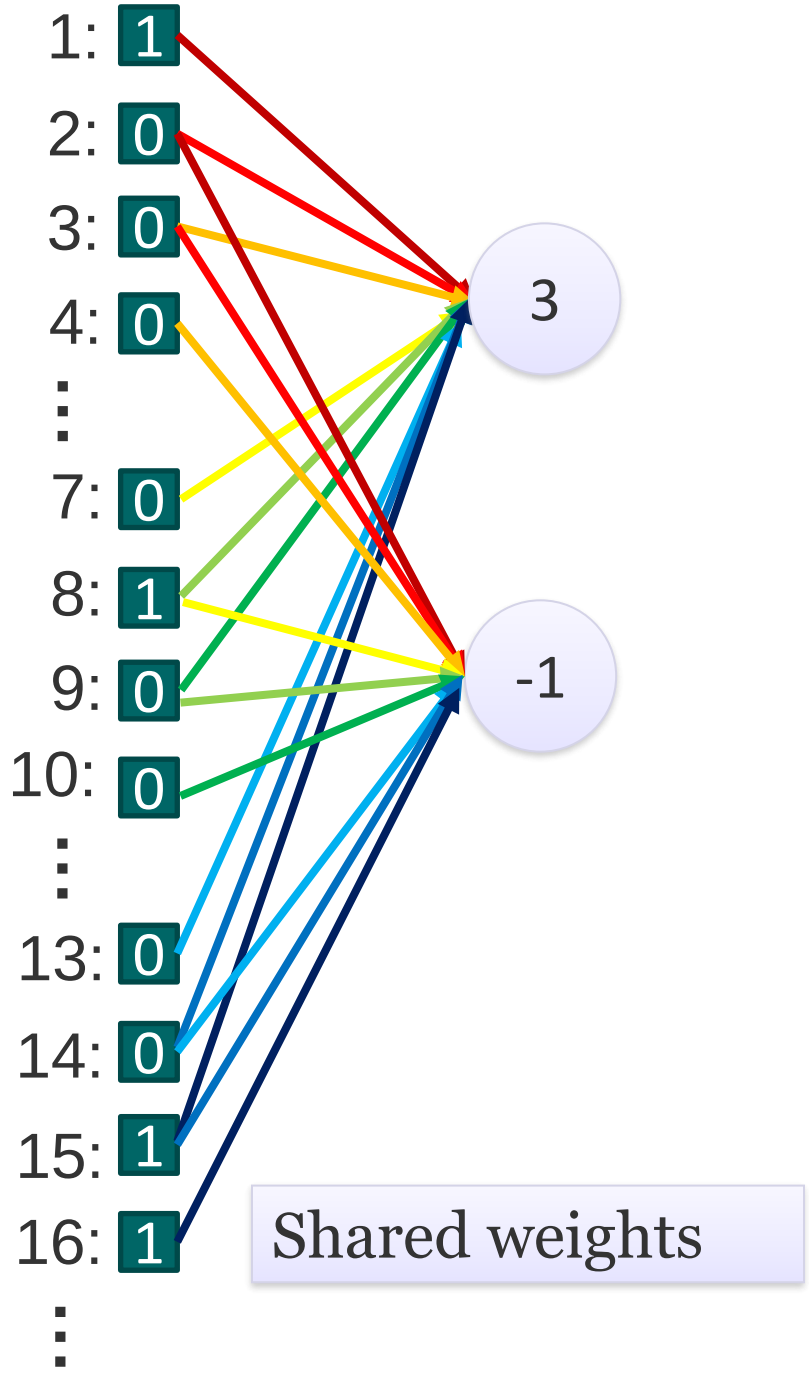


Fewer parameters!





Fewer parameters



Scale invariance: pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

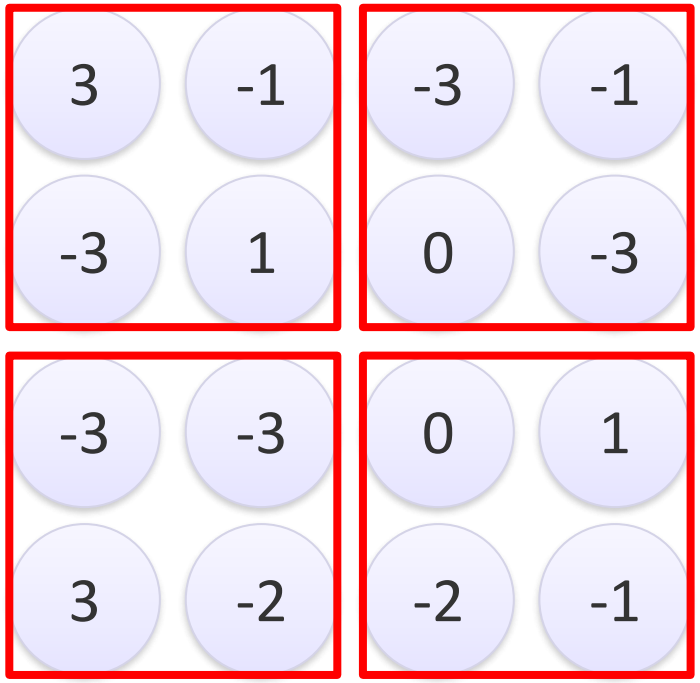
MaxPooling

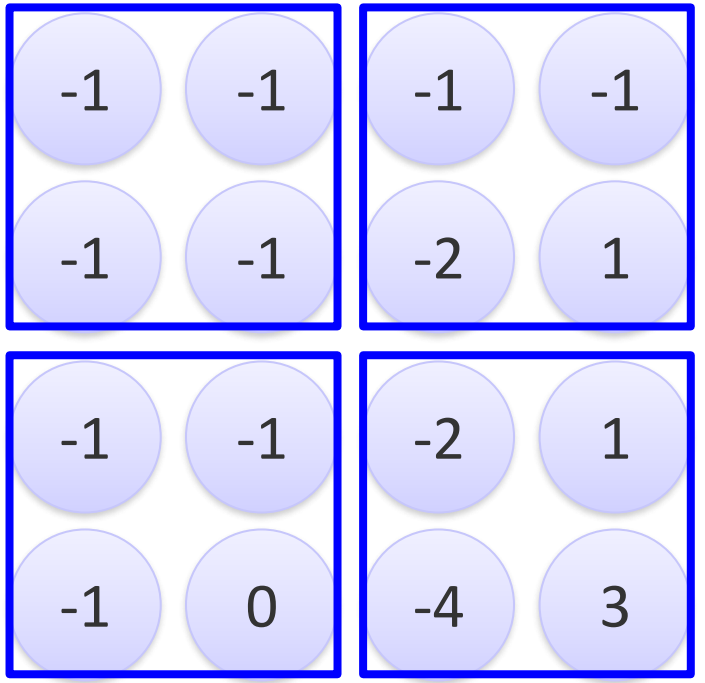
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

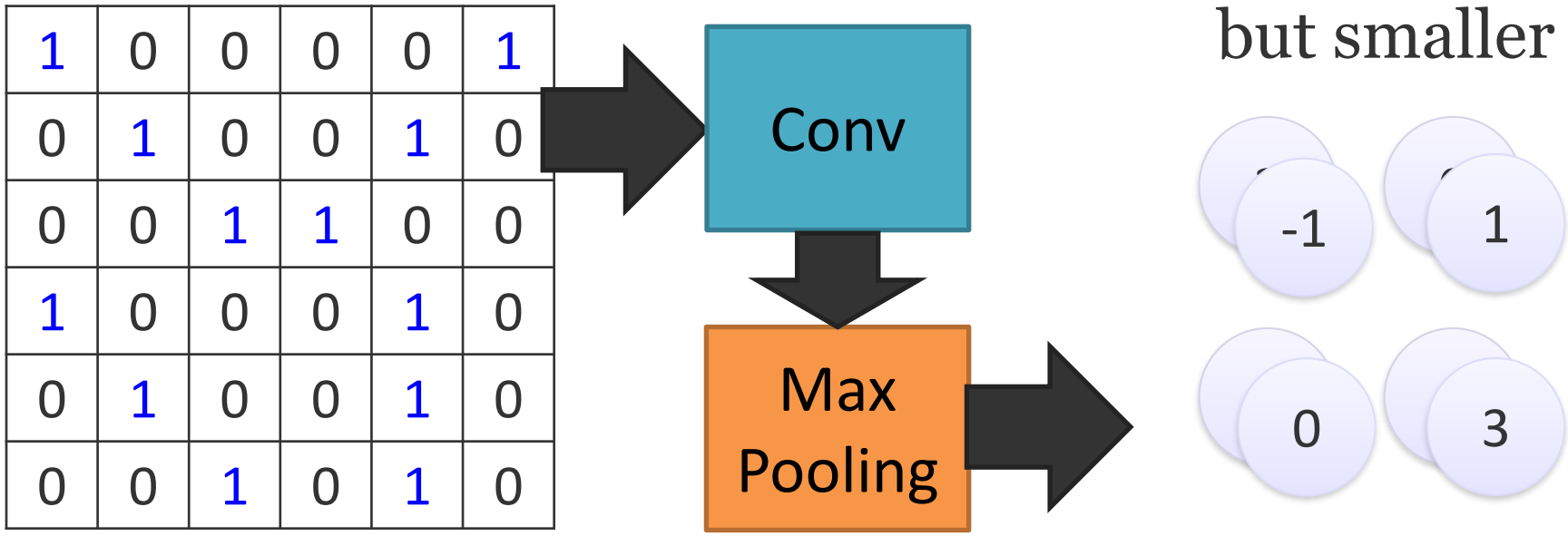
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2





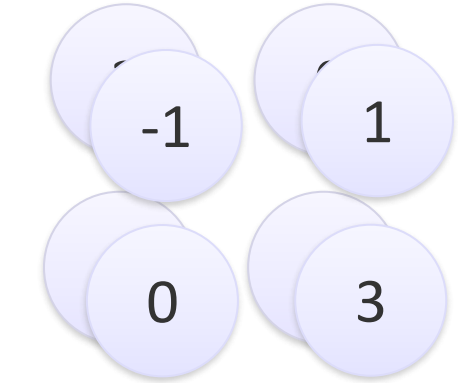
MaxPooling

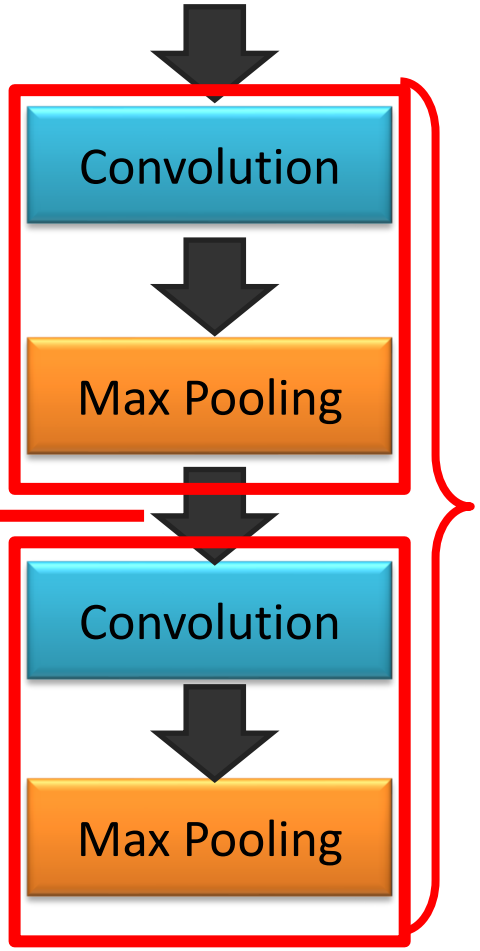


6 x 6 image

Each filter is a channel

Making DCNN





Can repeat many times

Smaller than the original image

The number of channels is the number of filters

Making DCNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

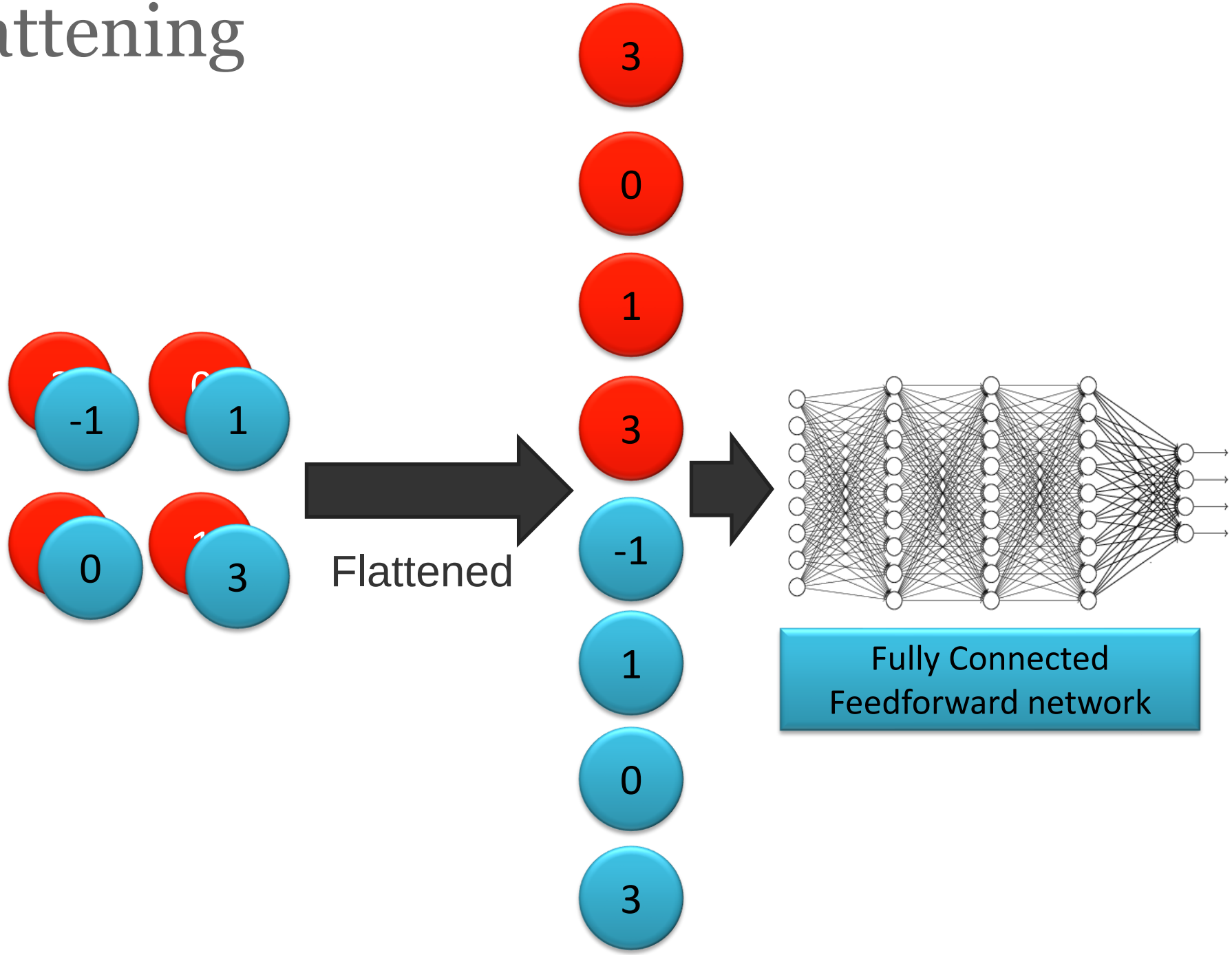
A new image

Flattened

Fully Connected
Feedforward network

cat dog ……

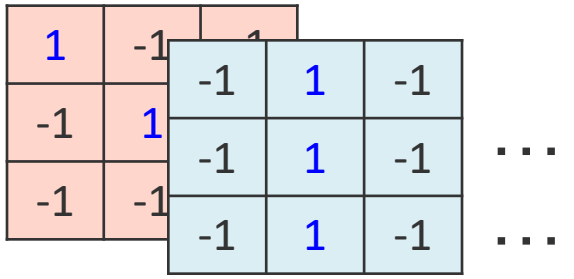
Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
                input_shape=(28,28,1)) )
```



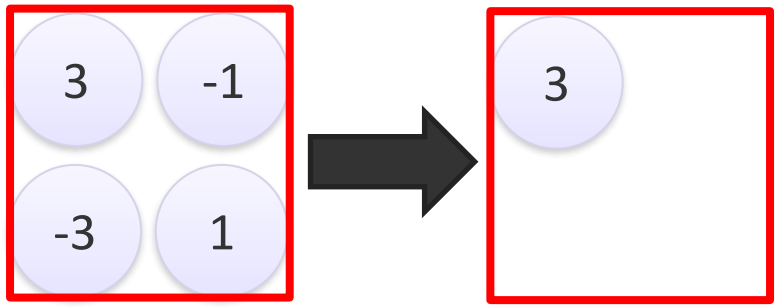
There are 25 3x3 filters.

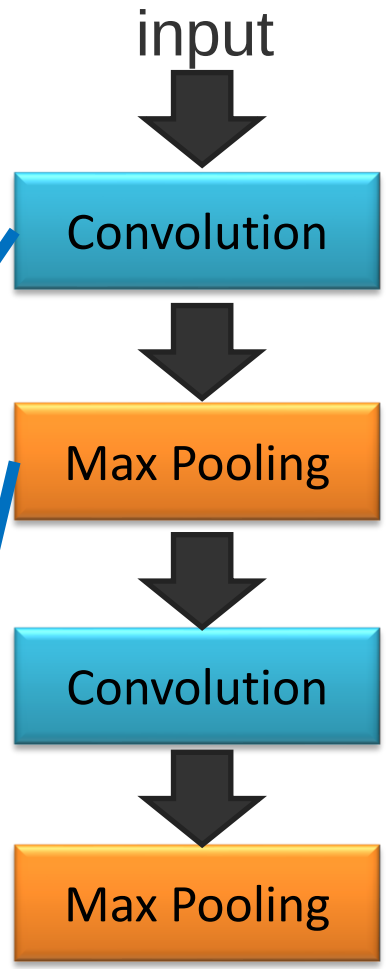
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

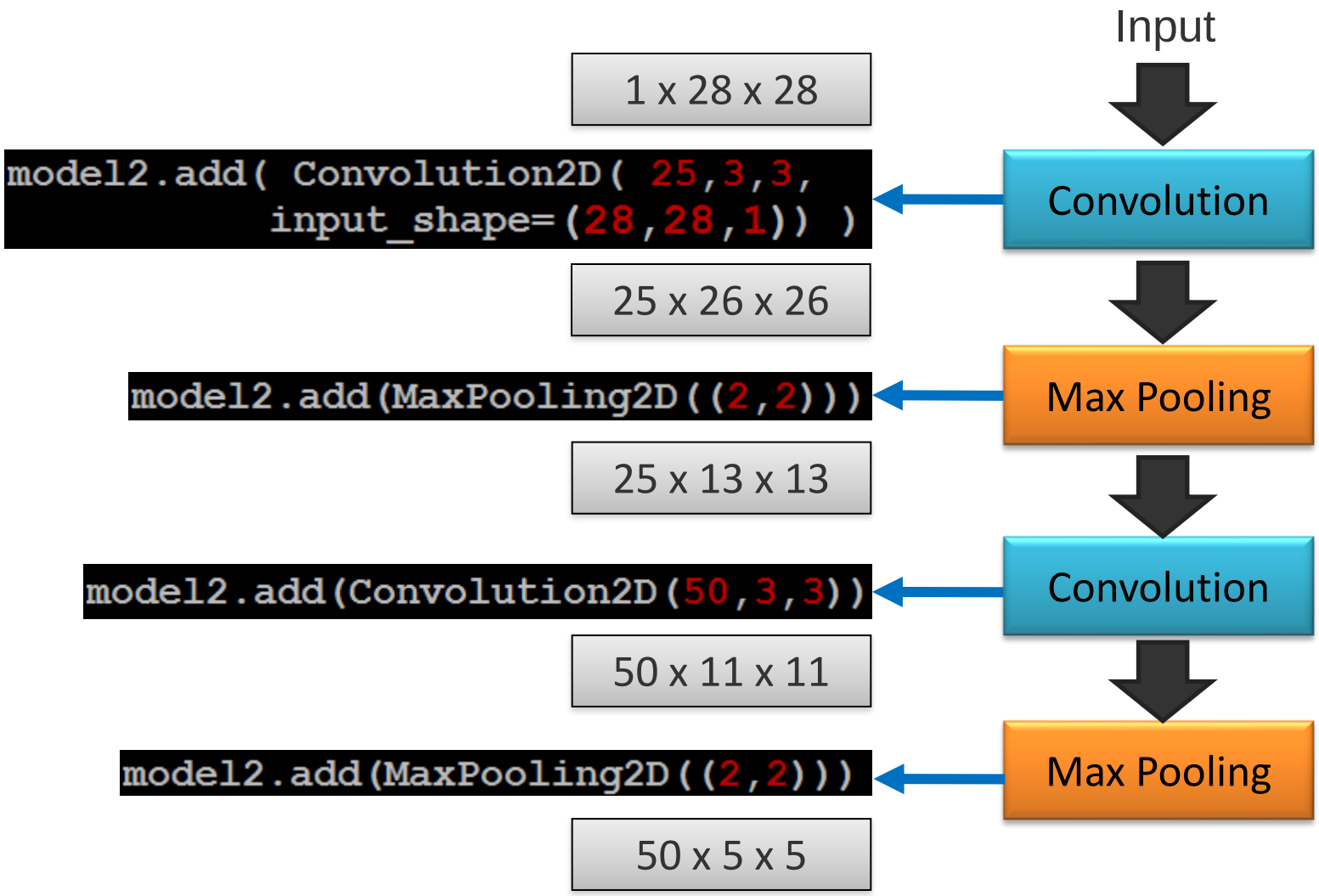
```
model2.add(MaxPooling2D((2,2)))
```





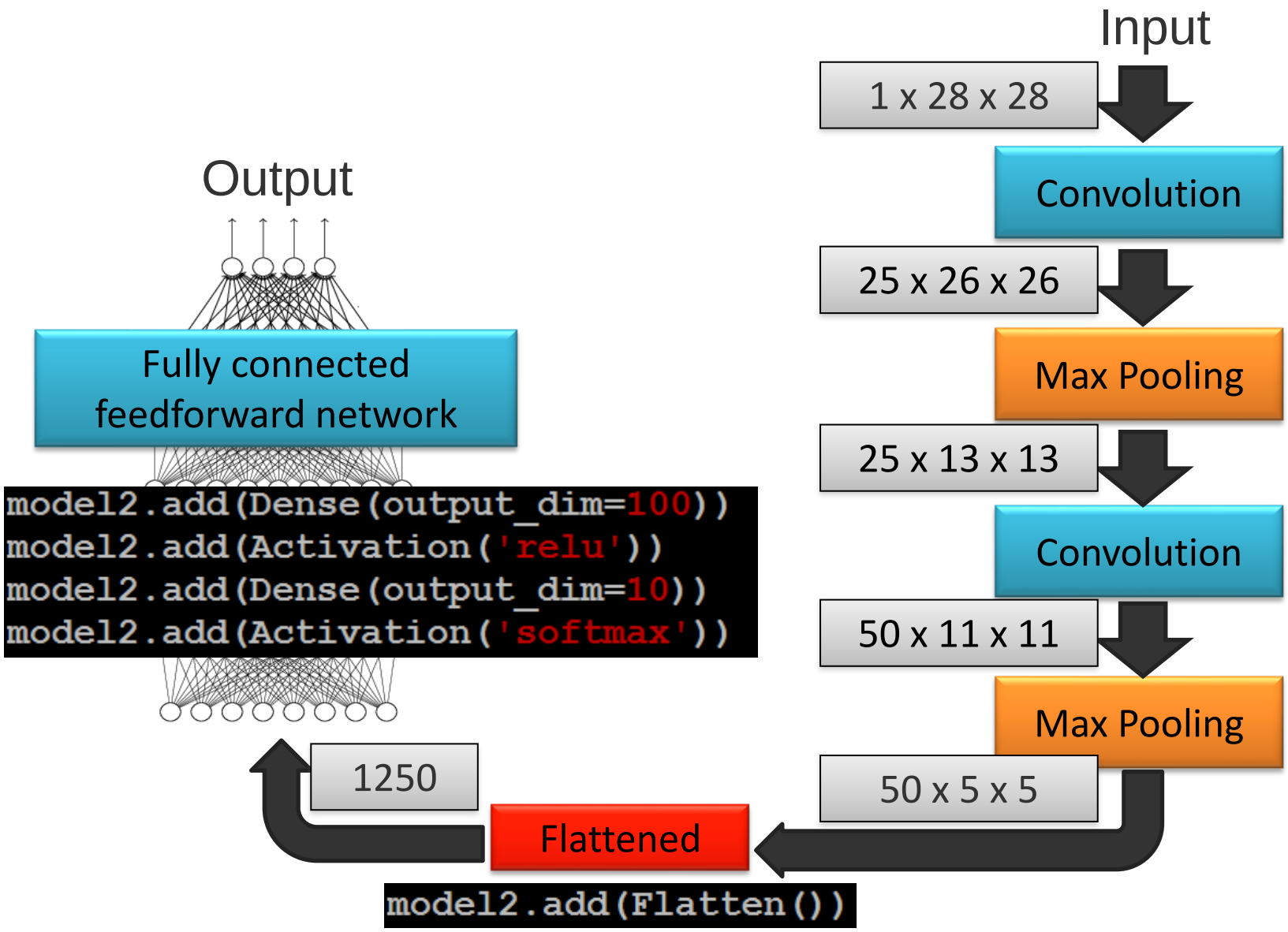
Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



Layer types

1. **Convolutional Layer:** Extracts features from the input image through convolution operations. Key Parameters: Number of filters, kernel size, strides, padding.

2. **Activation Layer:** Introduces non-linearity to the model, allowing it to learn complex patterns. Common Types: ReLU (Rectified Linear Unit), Sigmoid, Tanh.

3. **Pooling Layer:** Reduces the spatial dimensions (height and width) of the input volume, making the model more computationally efficient and less sensitive to the exact location of features. Types: Max Pooling, Average Pooling.

4. **Batch Normalization Layer:** Normalizes the output of a previous layer by subtracting the batch mean and dividing by the batch standard deviation. Helps in speeding up training and reducing the sensitivity to network initialization.

5. **Dropout Layer:** Randomly sets a fraction of input units to zero at each update during training, helping to prevent overfitting.

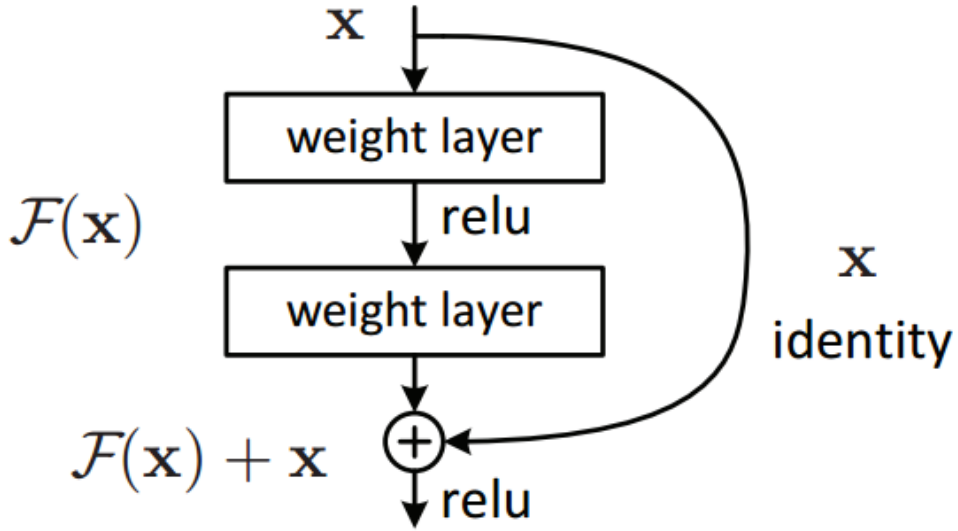
Layer types

6. **Fully Connected (Dense) Layer:** Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Used to classify or regress based on features extracted by the convolutional layers.

7. **Flatten Layer:** Flattens the input from a multi-dimensional tensor to a 1D tensor, making it possible to connect convolutional and pooling layers with dense layers.

8. **Residual Connections** (as in ResNet):Allows the output of one layer to bypass one or more layers and be added to the output of a later layer, improving gradient flow through the network.

9. **Transposed Convolutional Layer (Deconvolution):** Used in generative models and some segmentation tasks; it upscales the input feature map.



DCNNs vs. MLP

1. Parameter Efficiency

- Fewer Parameters: CNNs require significantly fewer parameters than FCNs. They use shared weights and convolutional filters, reducing the total number of trainable parameters. This makes CNNs more efficient and less memory-intensive.
- Reduces Overfitting: With fewer parameters, CNNs are less prone to overfitting, especially with image data.

2. Exploitation of Spatial Structure

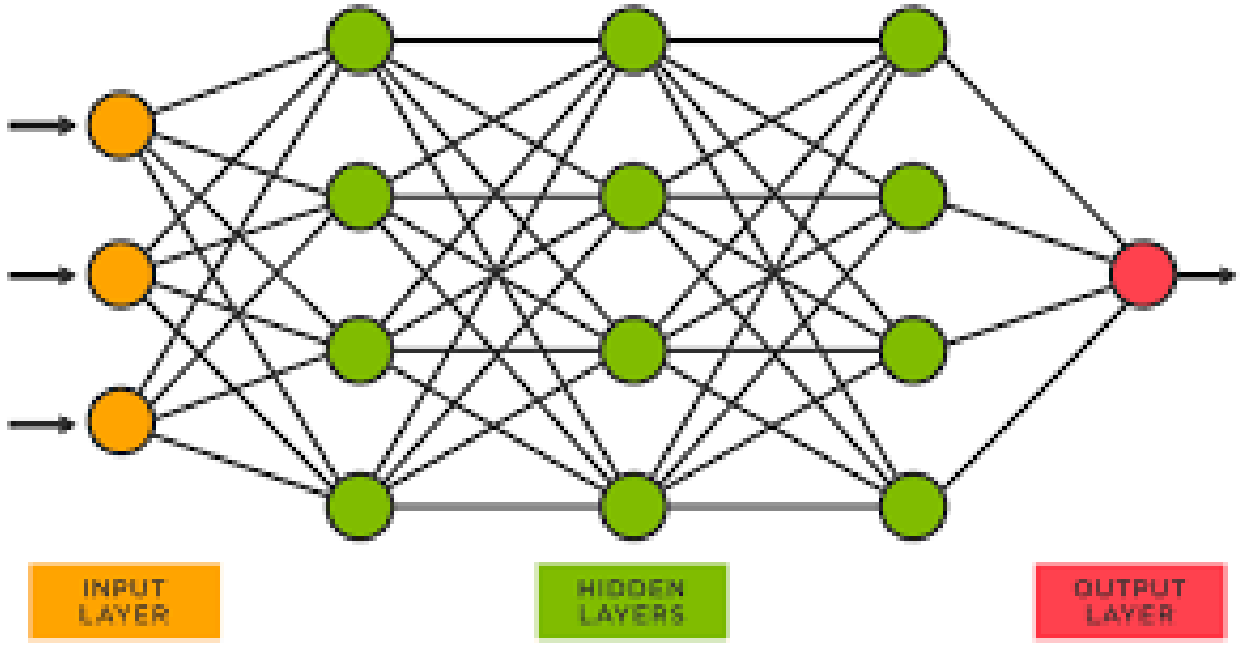
- Local Connectivity: CNNs exploit the spatial structure of the data by applying convolutional filters that capture local features (like edges, textures) in early layers and more complex features (like patterns or object parts) in deeper layers.
- Preservation of Spatial Relationships: Unlike FCNs that lose spatial relationships by flattening the input, CNNs maintain the spatial hierarchy and relationships between different parts of the input.

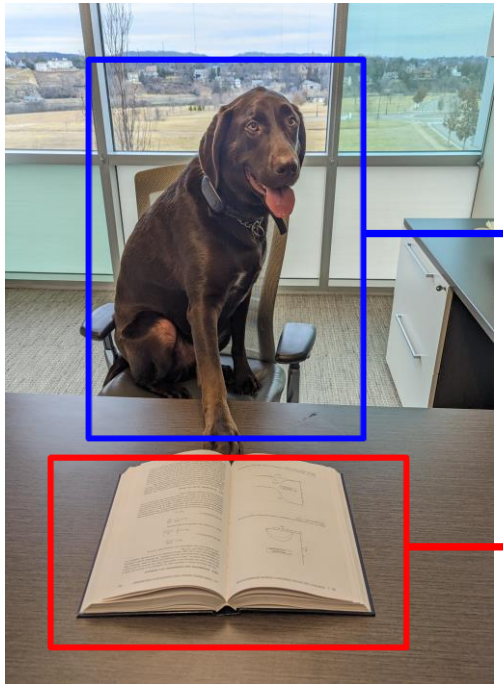
3. Translation Invariance

- Robust to Translation: Due to pooling layers and the nature of convolution operations, CNNs are inherently more robust to the translation of input data. This means that if an object shifts in an image, a CNN can still detect it effectively.

Supervised Machine Learning

- Regression
- Classification
- Semantic segmentation
- Instance segmentation
- …

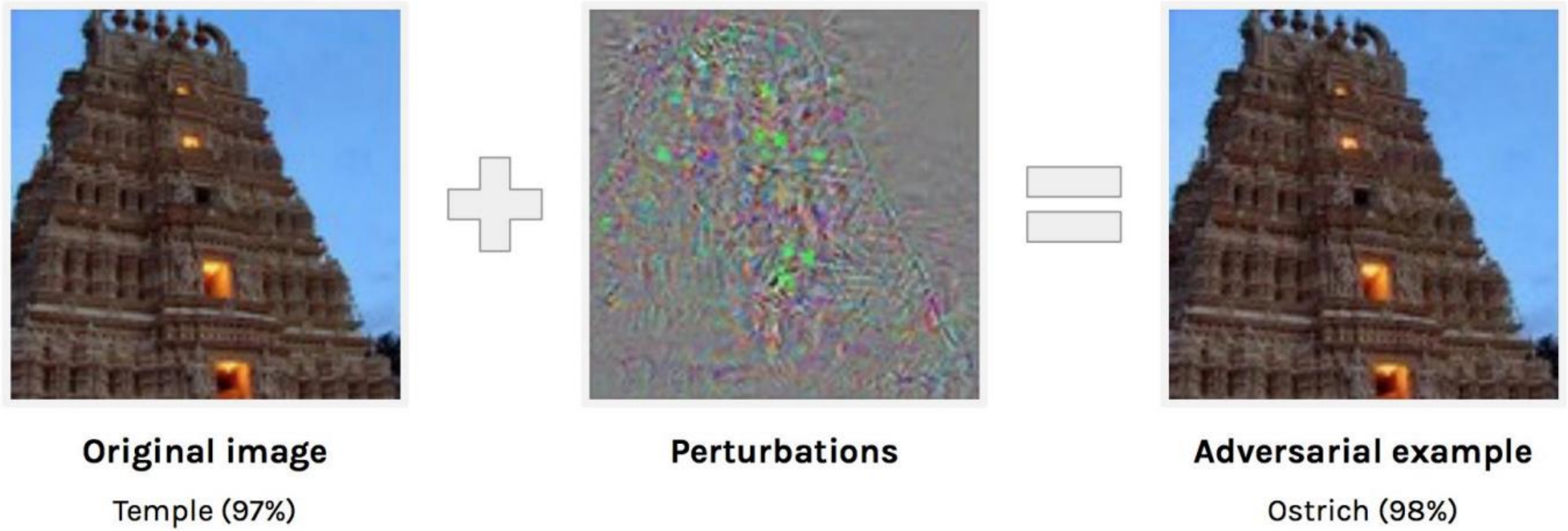




Dog

Book

Adversarial attacks



https://course.ccs.neu.edu/ds4420sp20/slides/18-ds4420-interpretability.pdf

What are the problems with ML models?

- We don’t trust the models
- We don’t know what happens in extreme cases
- Mistakes can be expensive / harmful
- Does the model make similar mistakes as humans ?
- How to change model when things go wrong ?

What do we want to get?

- Interactive feedback - can model learn from human actions in online setting ? (Can you tell a model to not repeat a specific mistake ?)
- Recourse – Can a model tell us what actions we can take to change its output ? (For example, what can you do to improve your credit score?)

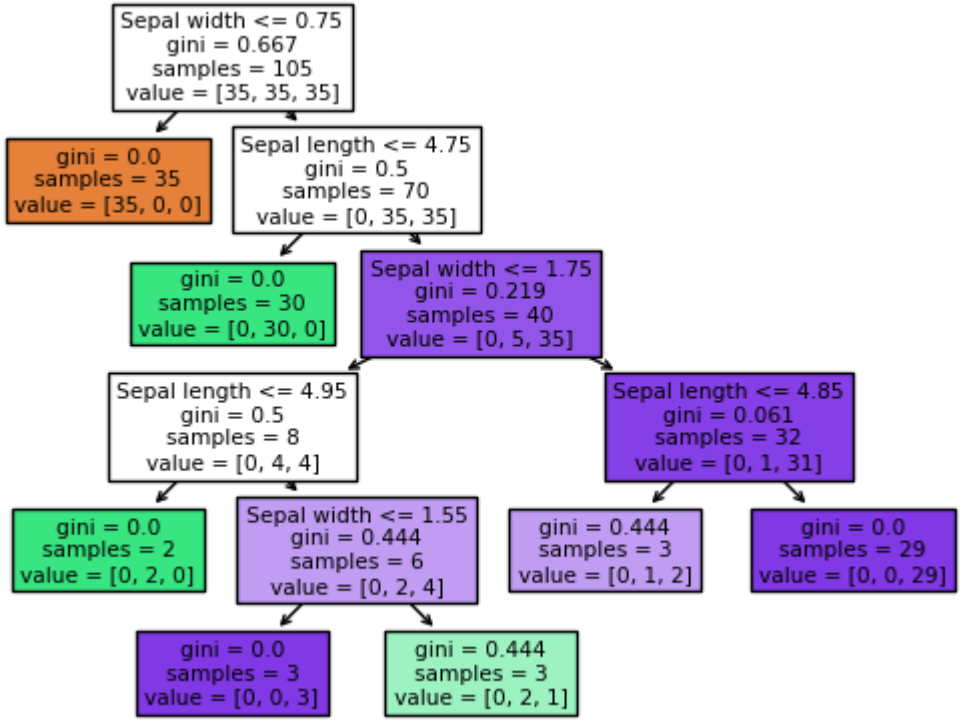
Models: Explainable and Not

Some models are explainable:

1. Linear or physics-defined function
2. Decision trees

But what about:

1. Image segmentation
2. Natural language processing
3. Classification
4. …



What is explainability?

- **Faithfulness:** how to provide explanations that accurately represent the true reasoning behind the model's final decision.
- **Plausibility:** Is the explanation correct or something we can believe is true, given our current knowledge of the problem
- **Understandable:** Can I put it in terms that end user without in-depth knowledge of the system can understand ?
- **Stability:** Does similar instances have similar interpretations ?

What do we expect from explainer?

1. Interpretable: It should provide a qualitative understanding between the input variables and the response. It should be easy to understand.

2. Local Fidelity: It might not be possible for an explanation to be completely faithful unless it is the complete description of the model itself. Having said that it should be at least locally faithful, i.e it must replicate the model's behavior in the vicinity of the instance being predicted.

3. Model Agnostic: The explainer should be able to explain any model and should not make any assumptions about the model while providing explanations.

4. Global perspective:The explainer should explain a representative set to the user so that the user has a global intuition of the model

https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5

Ways to explain ML methods

Global vs local:

- Do we explain individual prediction (Heatmaps, Rationales)?
- Do we explain entire model (Linear Regression, Decision Trees)?

Inherent or post-hoc:

- Is the explainability built into the model (Linear Regression, Decision Trees, Natural Language Explanations)
- Is the model black-box and we use external method to try to understand it (Heatmaps)?

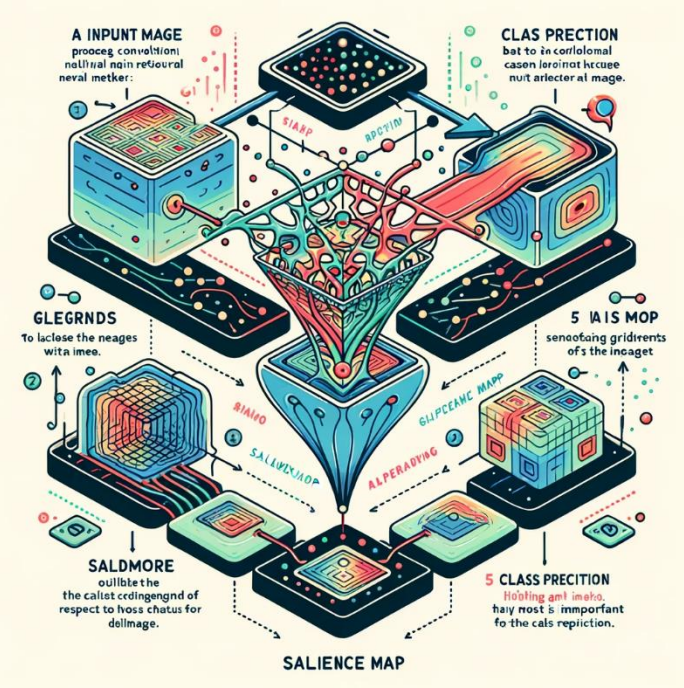
Model based vs Model Agnostic

- Can it explain only few classes of models (attention gradients – differentiable models only)
- Can it explain any model (LIME, SHAP)?

https://course.ccs.neu.edu/ds4420sp20/slides/18-ds4420-interpretability.pdf

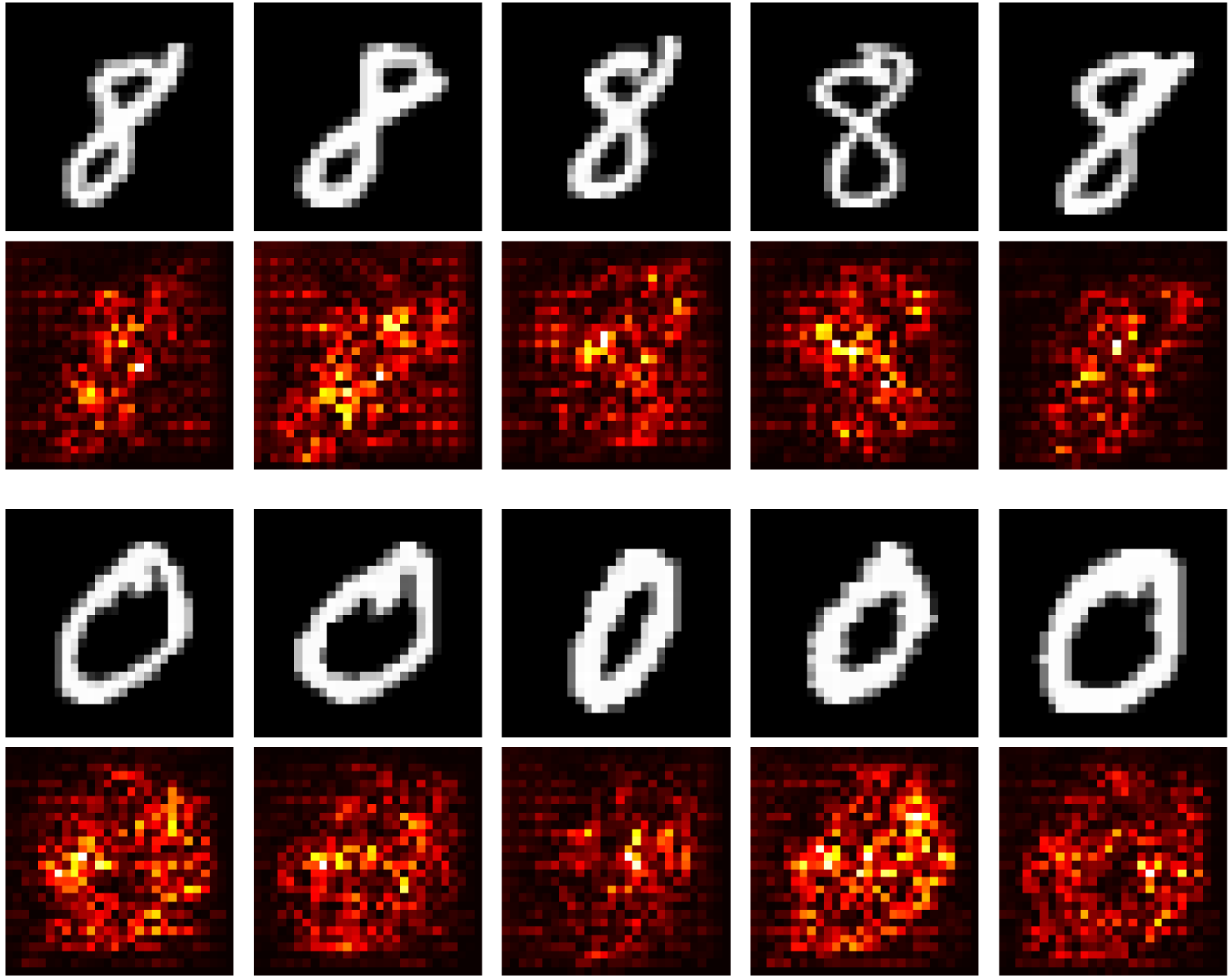
Saliency maps for differentiable models

- Choose the target class for which you want to compute the saliency map. This could be the class predicted by the network or any other class of interest.
- Pass the image through the model to get the output predictions. In the case of classification, this output is typically a probability distribution over classes.
- Extract the model's output (e.g., the probability or the logit) corresponding to the target class.

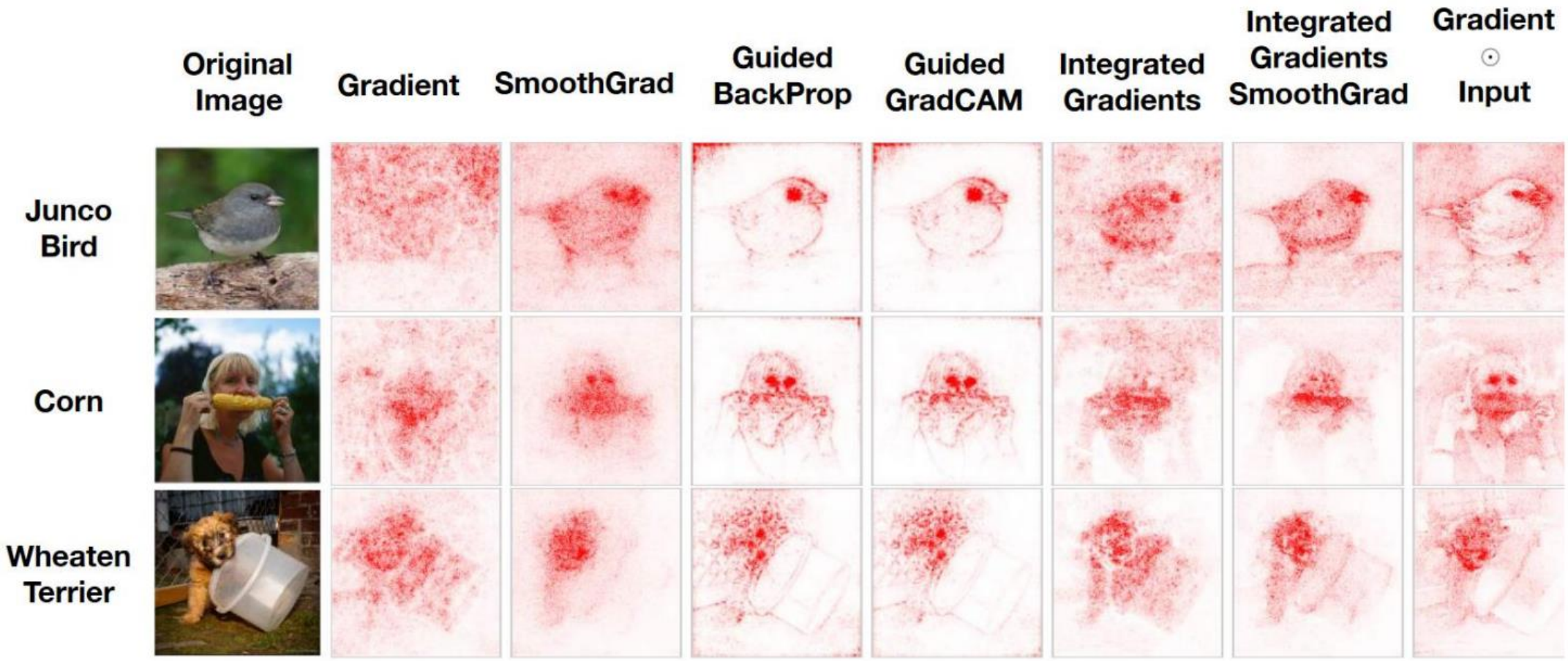


- Calculate the gradient of the output for the target class with respect to the input image. It highlights how much each pixel in the input image contributes to the output value for the chosen class.
- Post-process the gradients to create a saliency map:
  - Taking the absolute value of the gradient.
  - Collapsing the gradient across the color channels, often by taking the maximum or the average across channels.

Example of saliency maps for MNIST



There are many ways to get salience maps



[Adebayo et al 2018]

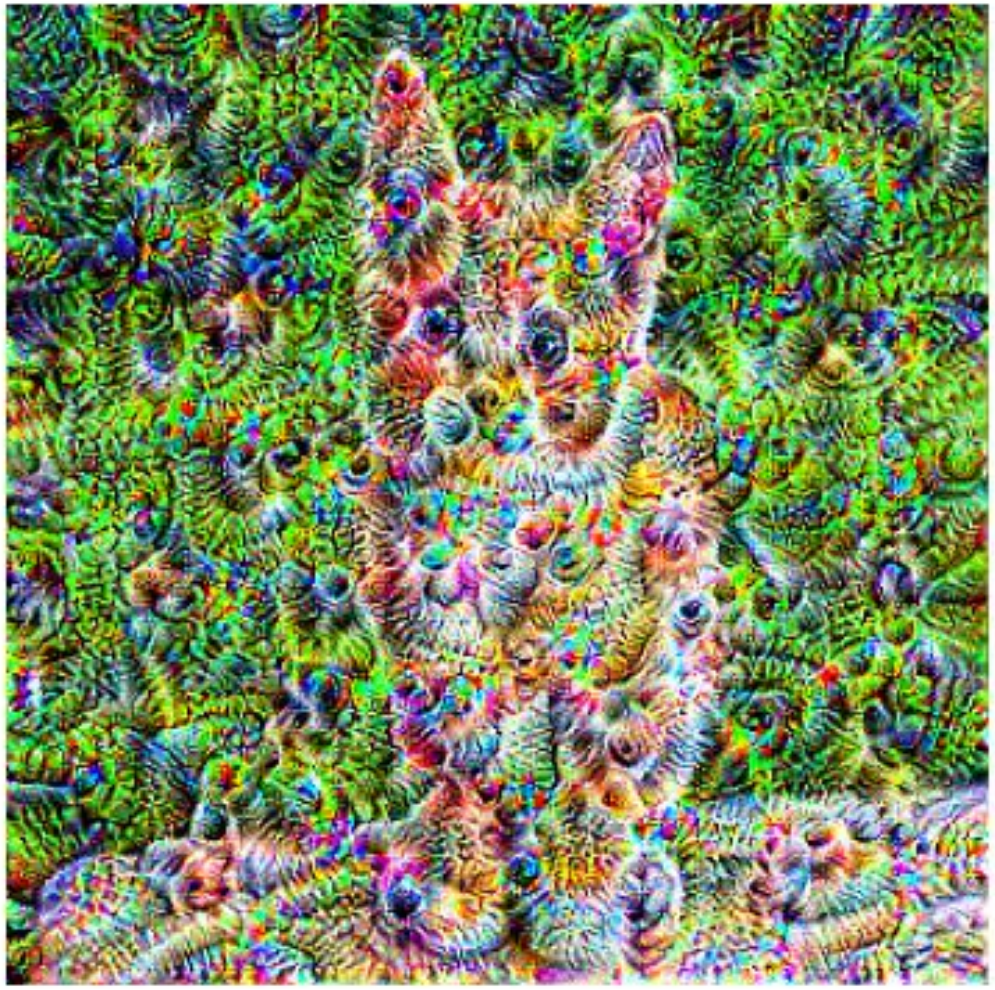
- Only capture first order information
- Not very reliable

https://course.ccs.neu.edu/ds4420sp20/slides/18-ds4420-interpretability.pdf

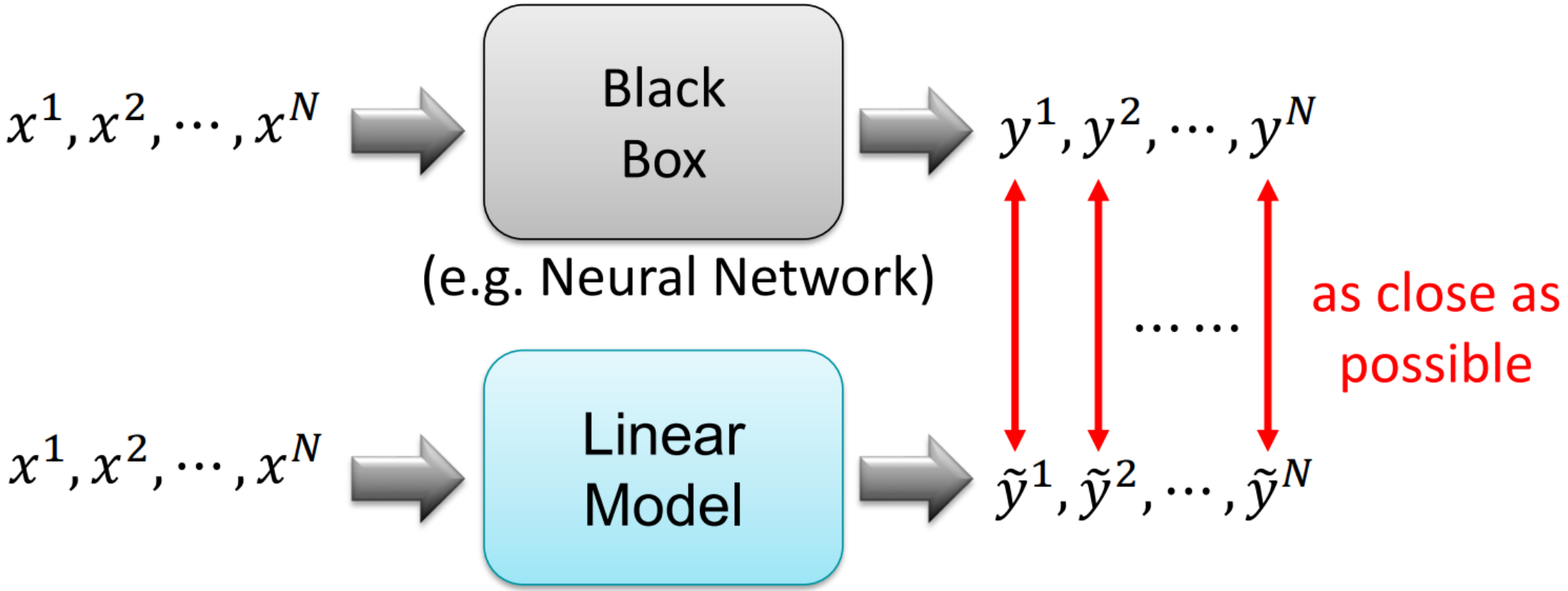
Closely related – Deep Dream

Original Image

Deep Dream Image



Locally Interpretable Manifold Embedding



https://course.ccs.neu.edu/ds4420sp20/slides/18-ds4420-interpretability.pdf

Locally Interpretable Manifold Embedding

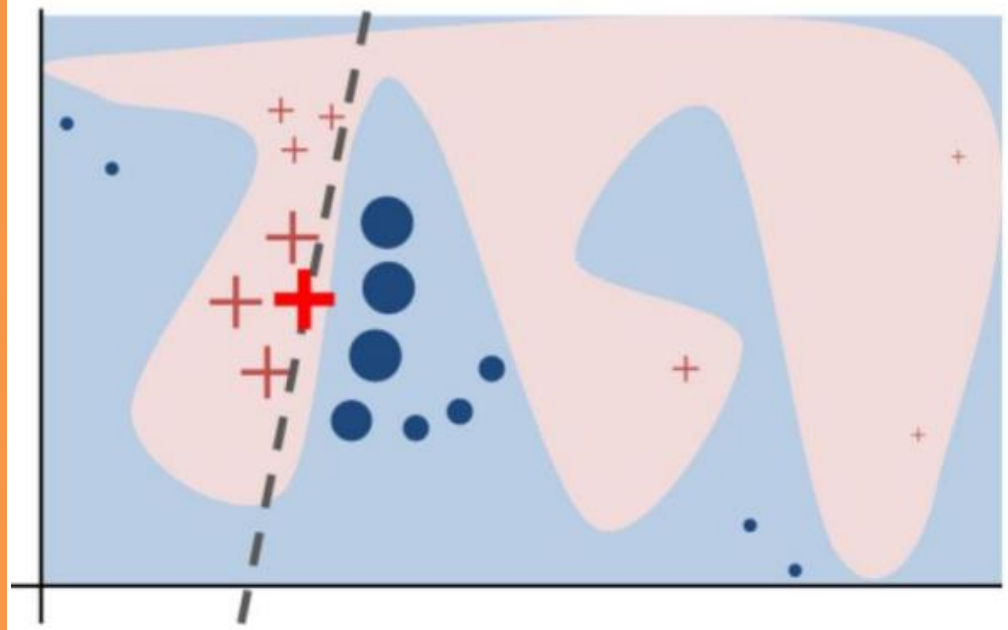
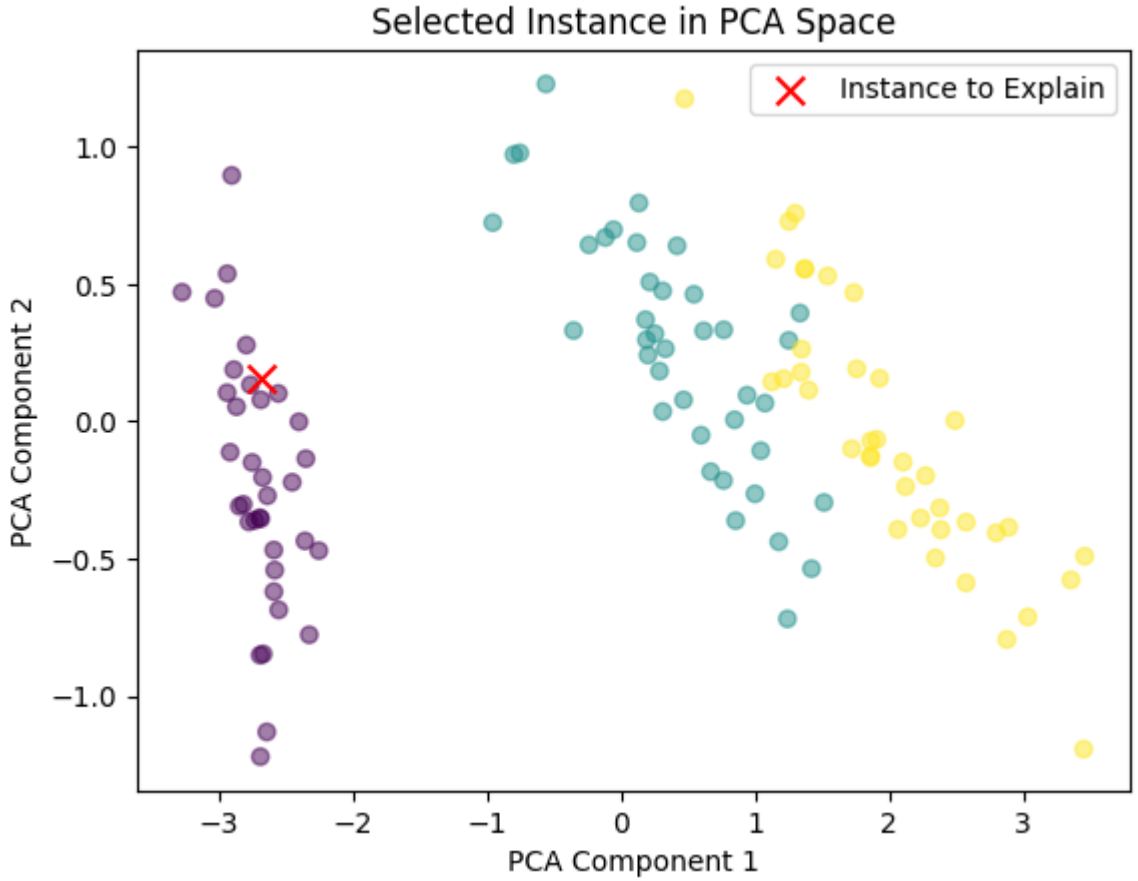


Figure 3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful.

Ribeiro et al, 2016

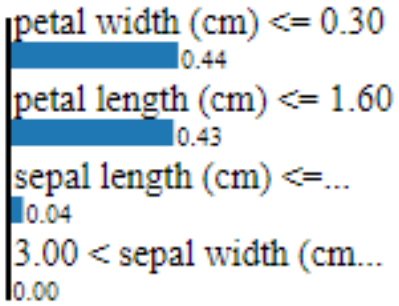
LIME on Iris



Prediction probabilities

setosa	1.00
versicolor	0.00
virginica	0.00

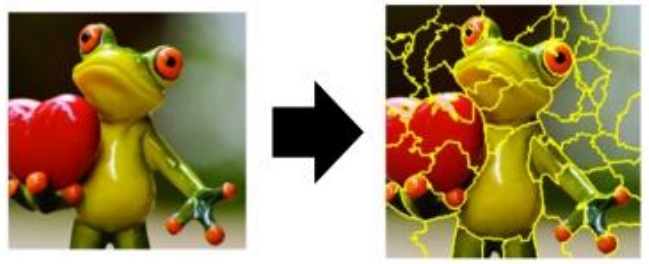
NOT setosa setosa



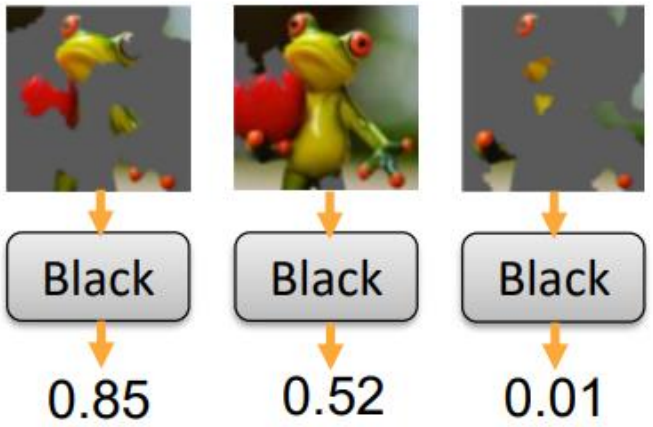
Feature	Value
petal width (cm)	0.20
petal length (cm)	1.60
sepal length (cm)	4.70
sepal width (cm)	3.20

LIME on Images

LIME – Image



- 1. Given a data point you want to explain
- 2. Sample at the nearby - Each image is represented as a set of superpixels (segments).



Randomly delete some segments.

Compute the probability of "frog" by black box

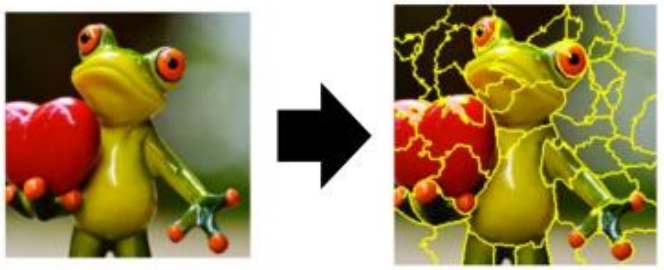
Ref: https://medium.com/@kstseng/lime-local-interpretable-model-agnostic-explanation%E6%8A%80%E8%A1%93%E4%BB%8B%E7%B4%B9-a67b6c34c3f8

(Slide Credit – Hung-yi Lee)

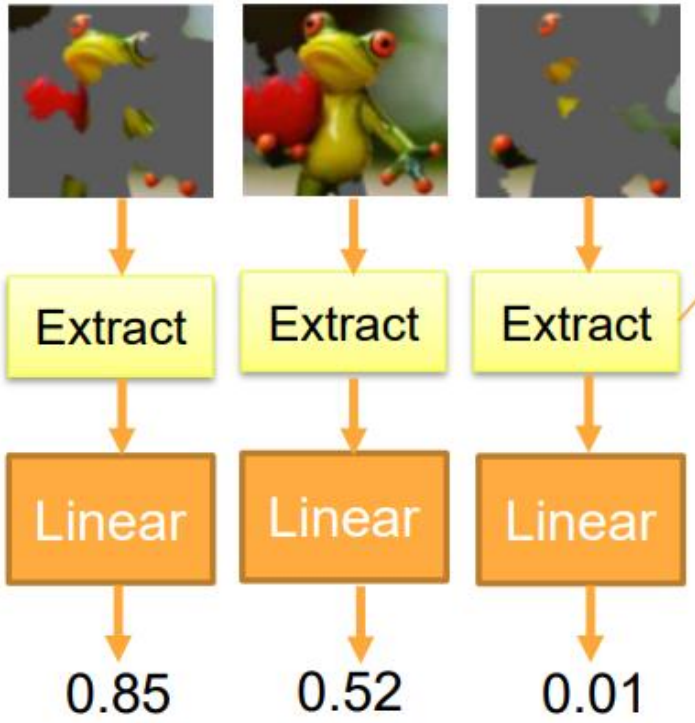
https://course.ccs.neu.edu/ds4420sp20/slides/18-ds4420-interpretability.pdf

LIME on Images

LIME – Image



- 3. Fit with linear (or interpretable) model



$$x_1 \quad \cdots\cdots \quad x_m \quad \cdots\cdots \quad x_M$$

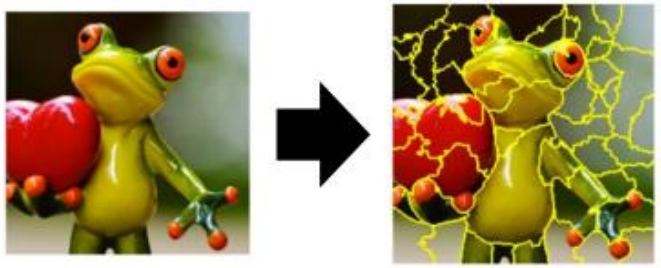
$$x_m = \begin{cases} 0 & \text{Segment m is deleted.} \\ 1 & \text{Segment m exists.} \end{cases}$$

M is the number of segments.

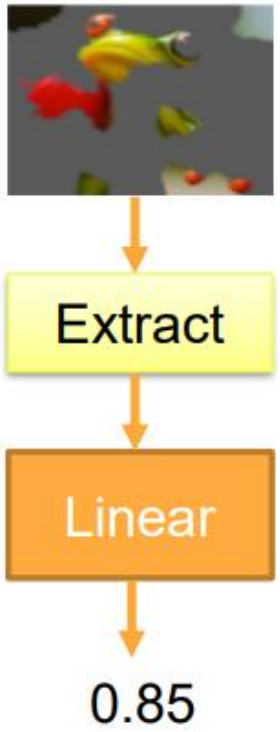
(Slide Credit – Hung-yi Lee)

LIME on Images

LIME – Image



- 4. Interpret the model you learned



$$y = w_1 x_1 + \cdots + w_m x_m + \cdots + w_M x_M$$

$$x_m = \begin{cases} 0 & \text{Segment m is deleted.} \\ 1 & \text{Segment m exists.} \end{cases}$$

M is the number of segments.

If $w_m \approx 0$ ➡ segment m is not related to "frog"

If w_m is positive ➡ segment m indicates the image is "frog"

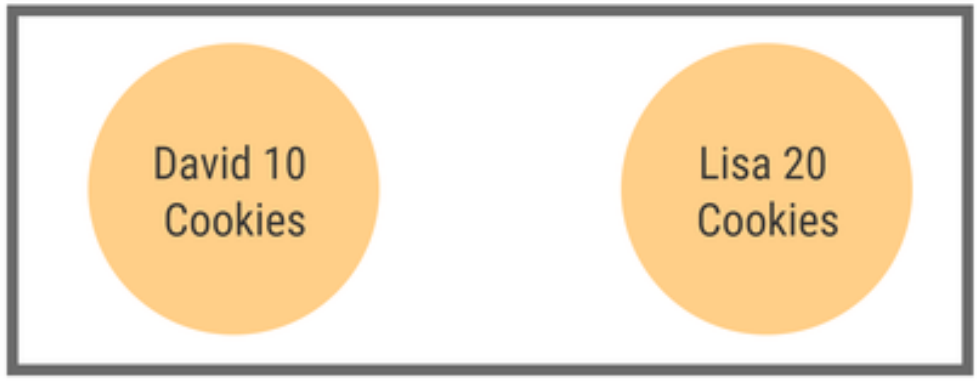
If w_m is negative ➡ segment m indicates the image is not "frog"

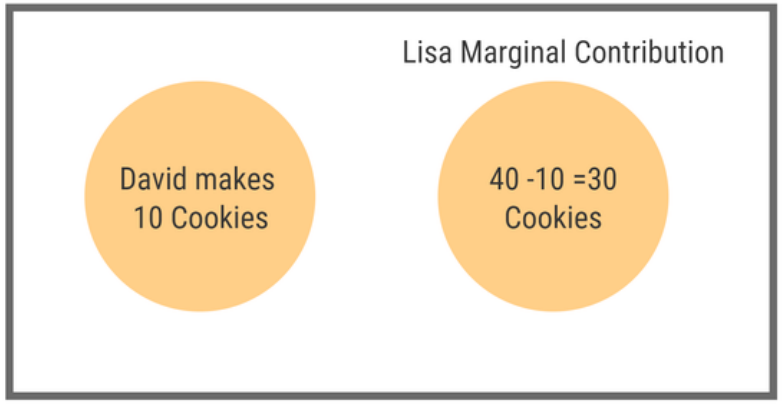
(Slide Credit – Hung-yi Lee)

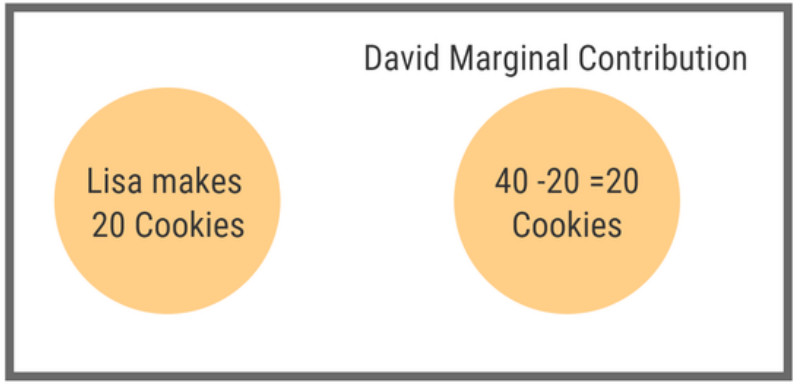
DYI LIME


Num features: 2
Num features: 3
Num features: 4
Num features: 5
Num features: 6
Num features: 7
Num features: 8
Num features: 9
Num features: 10
Num features: 11

SHAP: Shapley Additive exPlanations







- For Lisa, the contribution to the coalition is 30 cookies in the first case and her contribution to the coalition in the second case is 20 cookies. The Shapley value will be (20+30)/2 = 25
- To find the Shapley value of David, we need to average them: (10+20)/2 = 15.

https://abhishek-maheshwarappa.medium.com/shap-values-for-explainable-ai-58652645d881

SHAP: Shapley Additive exPlanations

A method of dividing up the gains or costs among players according to the value of their individual contributions.

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (v(S \cup \{i\}) - v(S))$$

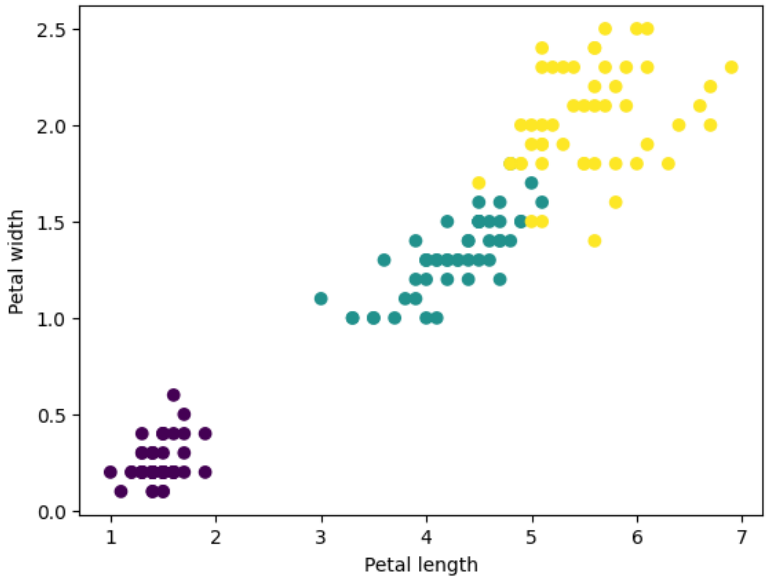
1. Marginal contribution
The contribution of each player is determined by what is gained or lost by removing them from the game. This is called their marginal contributions.

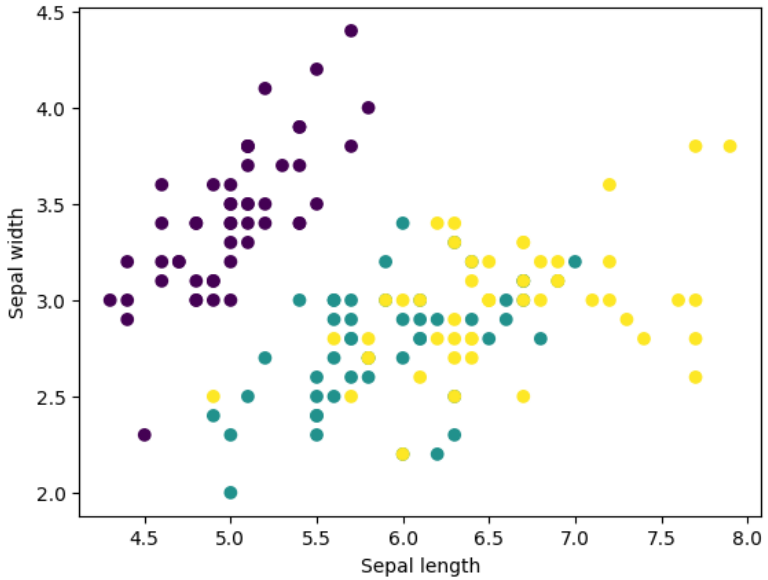
2. Interchangeable players have an equal value
If two parties bring the same things to the coalition, they should have to contribute the same amount and should be rewarded for their contributions.

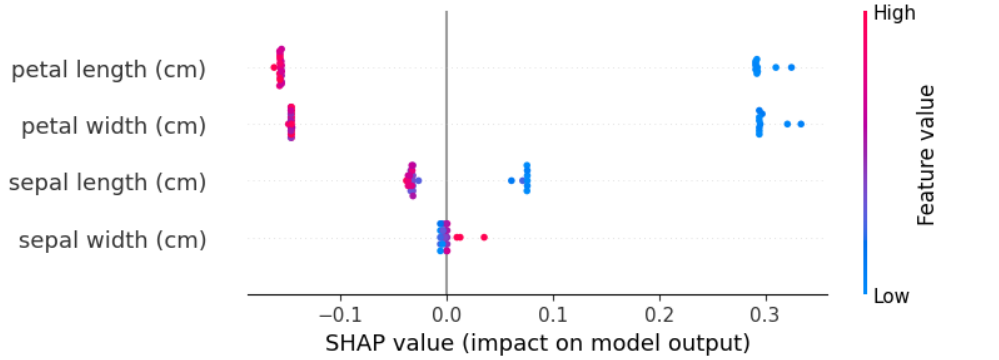
3. Dummy player has zero values
If a member of the coalition contributes nothing, then they should receive nothing. But it might not be fair in all cases, let us take an example of this thing more clear:

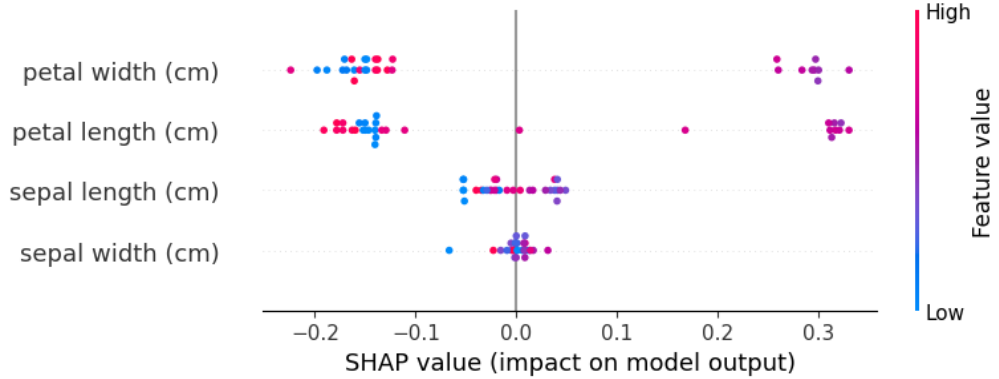
https://abhishek-maheshwarappa.medium.com/shap-values-for-explainable-ai-58652645d881

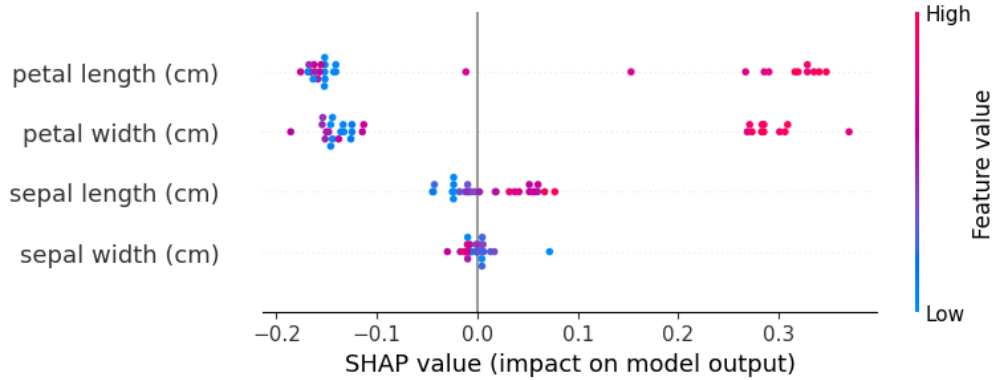
SHAP: Shapley Additive exPlanations



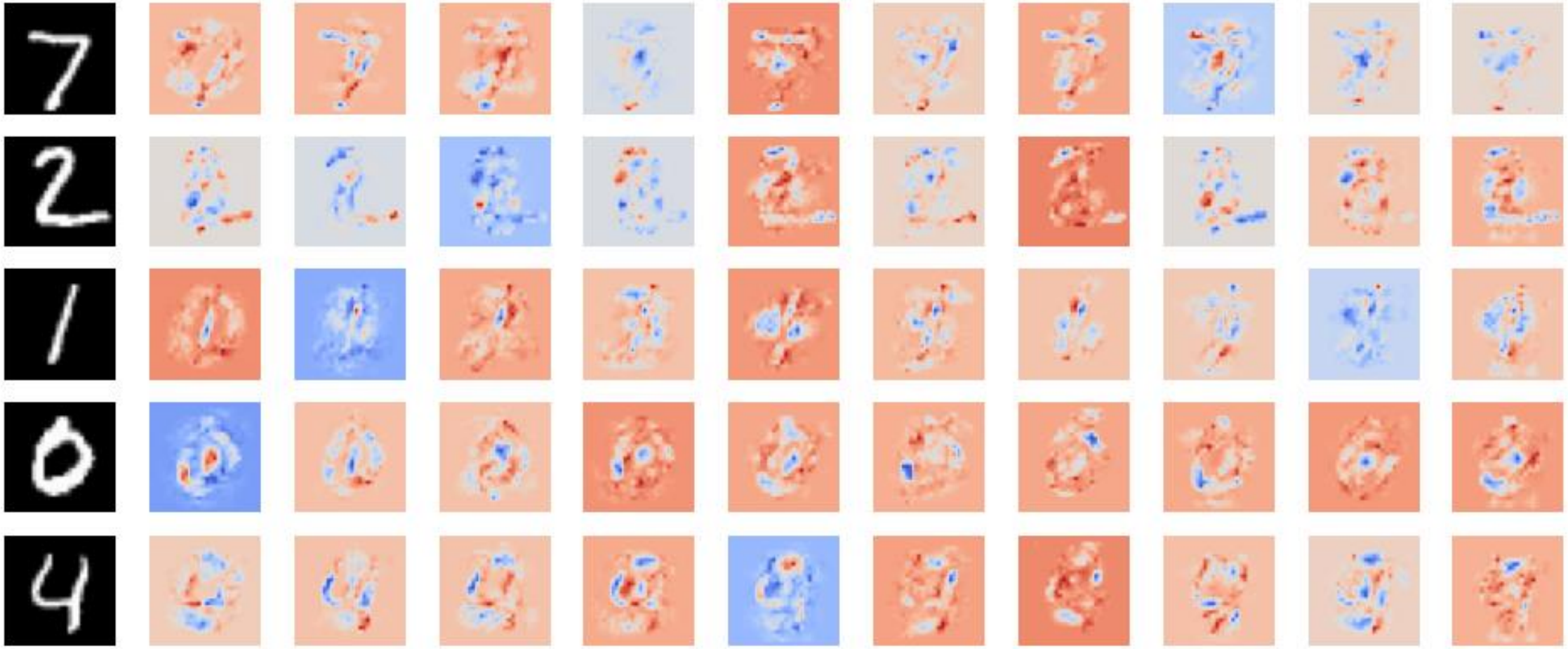


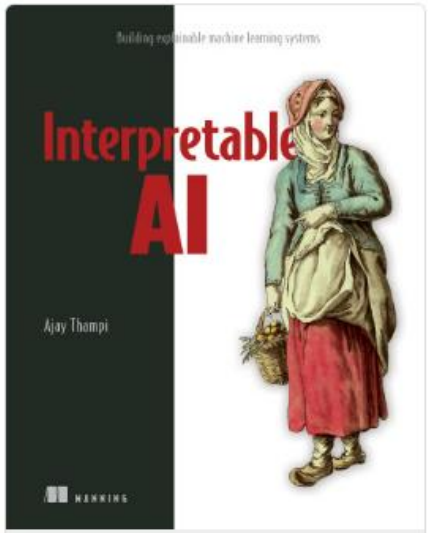






SHAP for Images





look inside

Transparent and understandable AI systems

read this article now in Manning's Free Content Center

resources

Source code

Book Forum

more

Interpretable AI you own this product

Building explainable machine learning systems

★★★★½ 6 reviews

Ajay Thampi

May 2022 · ISBN 978161729764-9 · 328 pages · printed in black & white

Data

eBook	print	subscription
pdf, ePub, online	includes eBook	from $19.99

AI doesn't have to be a black box. These practical techniques help shine a light on your model's mysterious inner workings. Make your AI more transparent, and you'll improve trust in your results, combat data leakage and bias, and ensure compliance with legal requirements.

In *Interpretable AI*, you will learn:

- Why AI models are hard to interpret
- Interpreting white box models such as linear regression, decision trees, and generalized additive models
- Partial dependence plots, LIME, SHAP and Anchors, and other techniques such as saliency mapping, network dissection, and representational learning
- What fairness is and how to mitigate bias in AI systems
- Implement robust AI systems that are GDPR-compliant

eBook

~~$47.99~~ **$31.19**

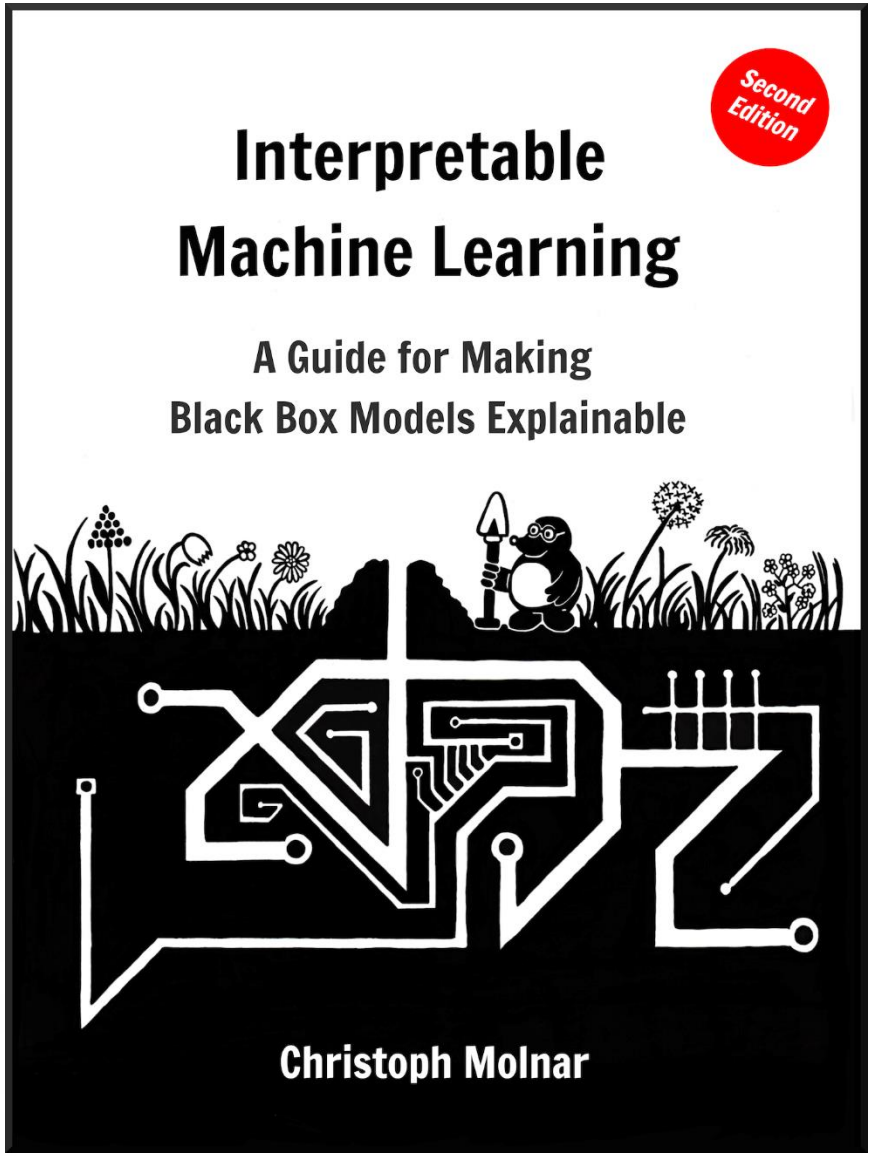
you save $16.80 (35%)

add to cart

buy now

free with subscription

> **A sound introduction for practitioners to the exciting field of interpretable AI.**
>
> Pablo Roccatagliata, Torcuato Di Tella University



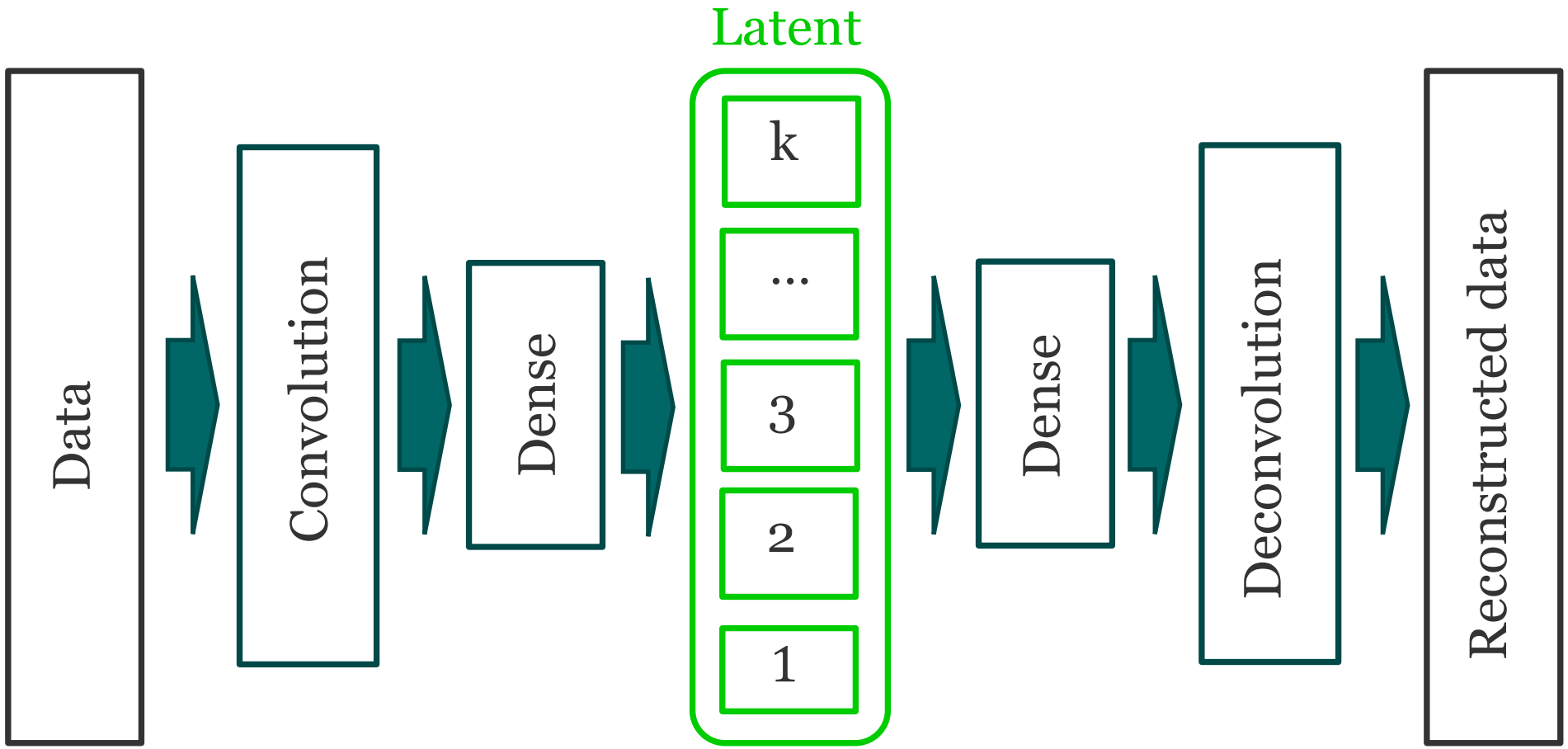
https://christophm.github.io/interpretable-ml-book/

Machine learning has great potential for improving products, processes and research. But **computers usually do not explain their predictions** which is a barrier to the adoption of machine learning. This book is about making machine learning models and their decisions interpretable.

After exploring the concepts of interpretability, you will learn about simple, **interpretable models** such as decision trees, decision rules and linear regression. The focus of the book is on model-agnostic methods for **interpreting black box models** such as feature importance and accumulated local effects, and explaining individual predictions with Shapley values and LIME. In addition, the book presents methods specific to deep neural networks.

All interpretation methods are explained in depth and discussed critically. How do they work under the hood? What are their strengths and weaknesses? How can their outputs be interpreted? This book will enable you to select and correctly apply the interpretation method that is most suitable for your machine learning project. Reading the book is recommended for machine learning practitioners, data scientists, statisticians, and anyone else interested in making machine learning models interpretable.

Autoencoders

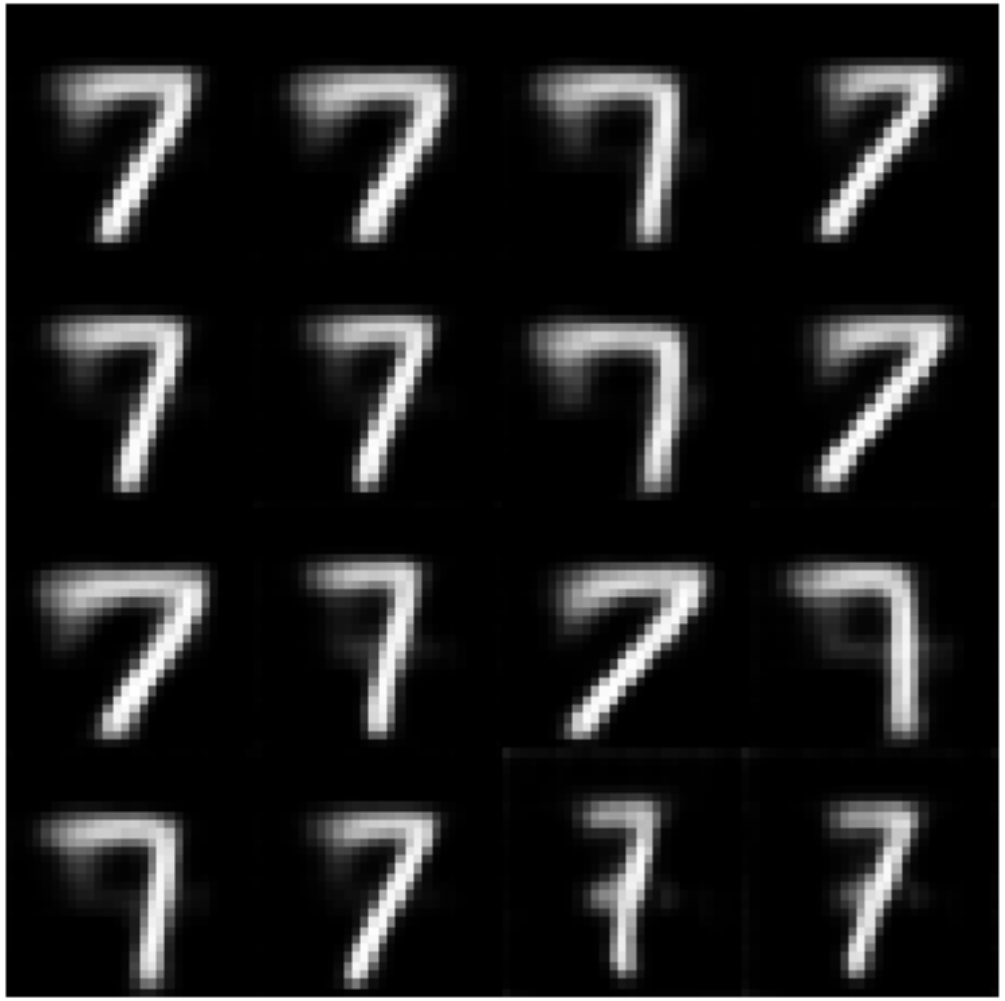


Loss: reconstruction loss

The AE reconstructs data

Input data

Decoded data



Why are AE important?

Geoffrey Hinton

FOLLOW

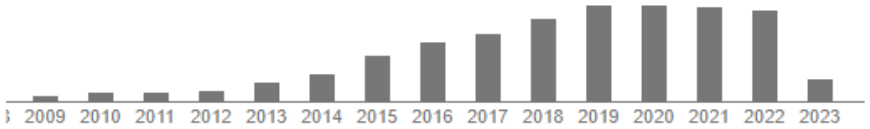
Emeritus Prof. Comp Sci, U.Toronto & Engineering Fellow, Google
Verified email at cs.toronto.edu - Homepage

machine learning psychology artificial intelligence cognitive science computer science

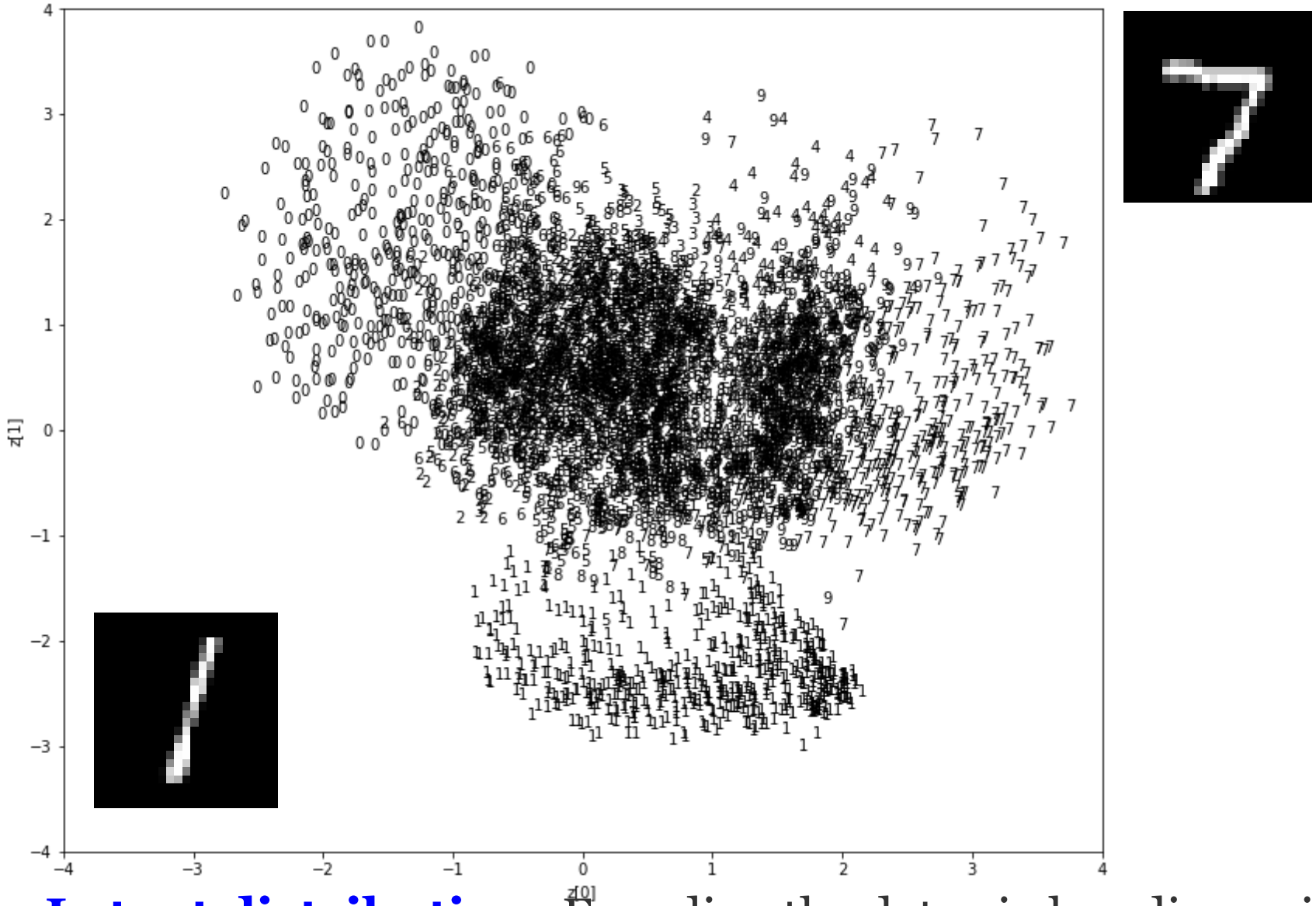
TITLE	CITED BY	YEAR
Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton Communications of the ACM 60 (6), 84-90	130318	2017
Deep learning Y LeCun, Y Bengio, G Hinton Nature 521 (7553), 436-44	62790	2015
Dropout: a simple way to prevent neural networks from overfitting N Srivastava, G Hinton, A Krizhevsky, I Sutskever, R Salakhutdinov The journal of machine learning research 15 (1), 1929-1958	42078	2014
Visualizing data using t-SNE L van der Maaten, G Hinton Journal of Machine Learning Research 9 (Nov), 2579-2605	35035	2008
Learning representations by back-propagating errors DE Rumelhart, GE Hinton, RJ Williams Nature 323 (6088), 533-536	32239	1986
Learning internal representations by error-propagation DE Rumelhart, GE Hinton, RJ Williams Parallel Distributed Processing: Explorations in the Microstructure of ...	30711	1986
Schemata and sequential thought processes in PDP models. D Rumelhart, P Smolenksy, J McClelland, G Hinton Parallel distributed processing: Explorations in the microstructure of ...	28073 *	1986
Learning multiple layers of features from tiny images A Krizhevsky, G Hinton	21876	2009
Rectified linear units improve restricted boltzmann machines V Nair, GE Hinton Proceedings of the 27th international conference on machine learning (ICML ...	21050	2010
Reducing the dimensionality of data with neural networks GE Hinton, RR Salakhutdinov Science 313 (5786), 504-507	19930	2006

Reducing the dimensionality of data with neural networks

Authors	Geoffrey E Hinton, Ruslan R Salakhutdinov
Publication date	2006/7/28
Journal	Science
Volume	313
Issue	5786
Pages	504-507
Publisher	American Association for the Advancement of Science
Description	High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.
Total citations	Cited by 19930

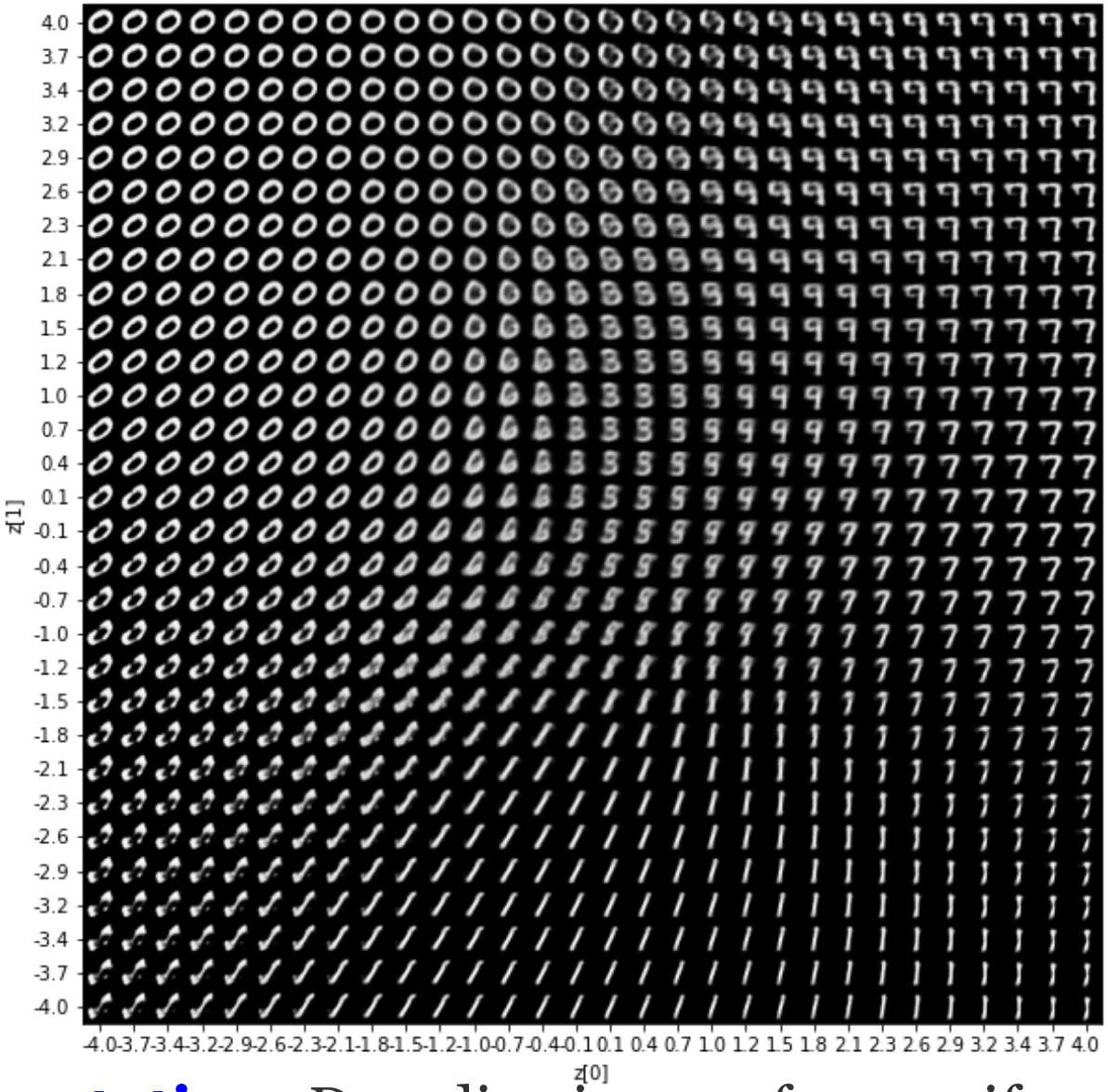


Encoding: Image → Latent Space



Latent distribution: Encoding the data via low dimensional vector

Decoding: Latent Space → Image



Latent representation: Decoding images from uniform grid in latent space

Image Reconstruction

Test color images (Ground Truth)

Test gray images (Input)

Image Reconstruction

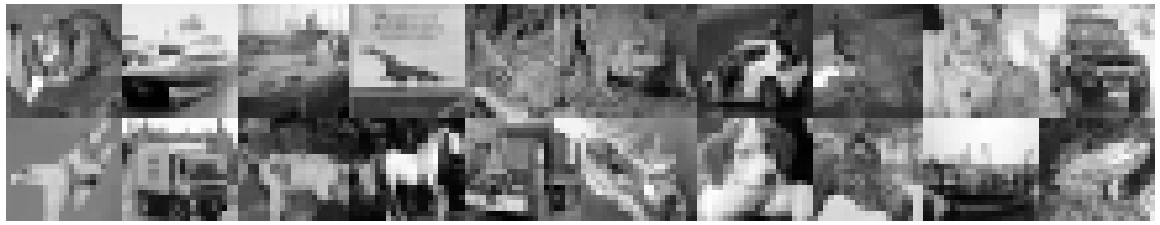
Test color images (Ground Truth)

Colorized test images (Predicted)

Image Reconstruction

Test color images (Ground Truth)

Test gray images (Input)



Colorized test images (Predicted)

- **Training:** pairs of the grayscale and color images
- **Application:** new grayscale images (from the same distribution)
- **Concern:** has to be from the same distribution

Image Denoising

Ground truth

Noisy input

Reconstruction

- **Training:** pairs of the high-noise and low-noise images
- **Application:** new high noise images (from the same distribution)
- **Concern:** has to be from the same distribution

Variational Autoencoders

Diederik P. Kingma

FOLLOW

Other names ▸

Research Scientist, Google Brain
Verified email at google.com - Homepage

Machine Learning Deep Learning Neural Networks Generative Models Variational Inference

TITLE	CITED BY	YEAR
Adam: A Method for Stochastic Optimization DP Kingma, J Ba Proceedings of the 3rd International Conference on Learning Representations ...	141306	2014
Auto-Encoding Variational Bayes DP Kingma, M Welling arXiv preprint arXiv:1312.6114	26540	2013
Semi-Supervised Learning with Deep Generative Models DP Kingma, S Mohamed, DJ Rezende, M Welling Advances in Neural Information Processing Systems, 3581-3589	2946	2014

- Variational Autoencoder (VAE): uses “reparameterization trick” to sample from the latent space
- Can be used for same tasks as AE
- Have a much better-behaved latent space: disentanglement of the representations

VAE Training

Latent manifold → Image space

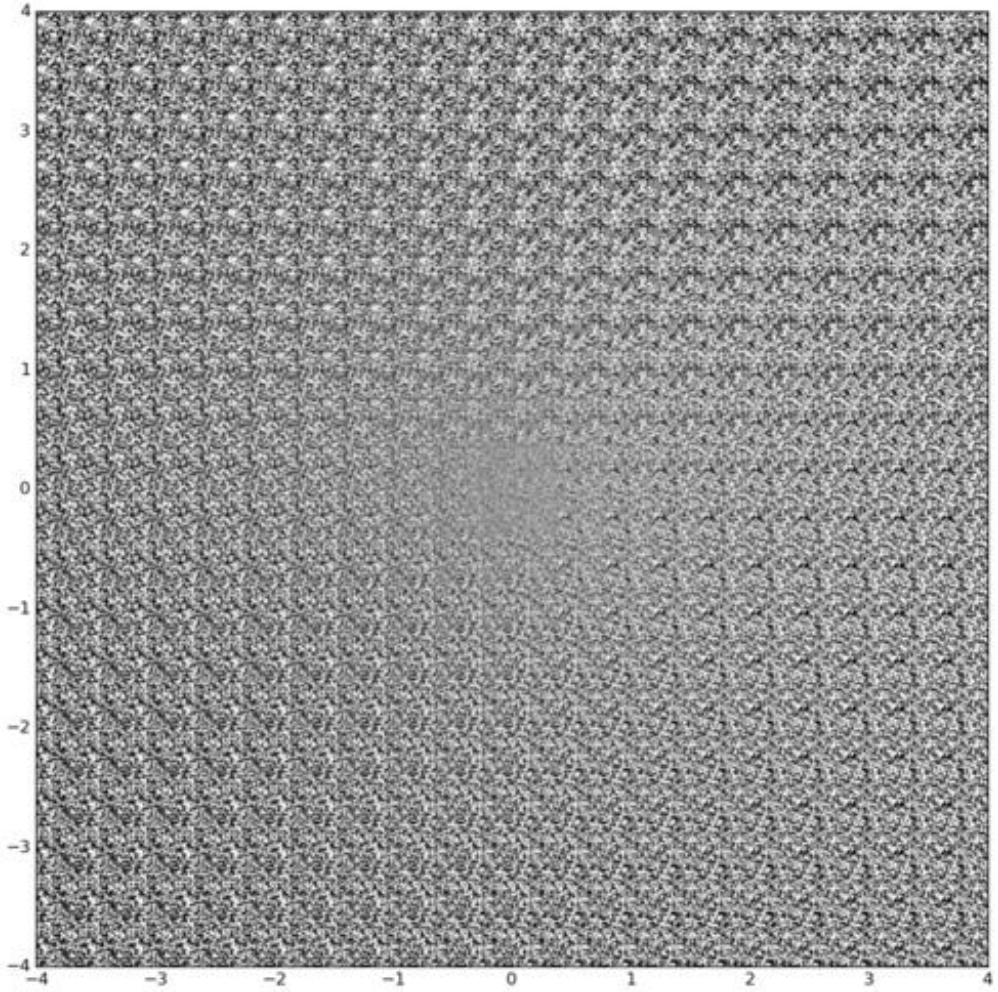
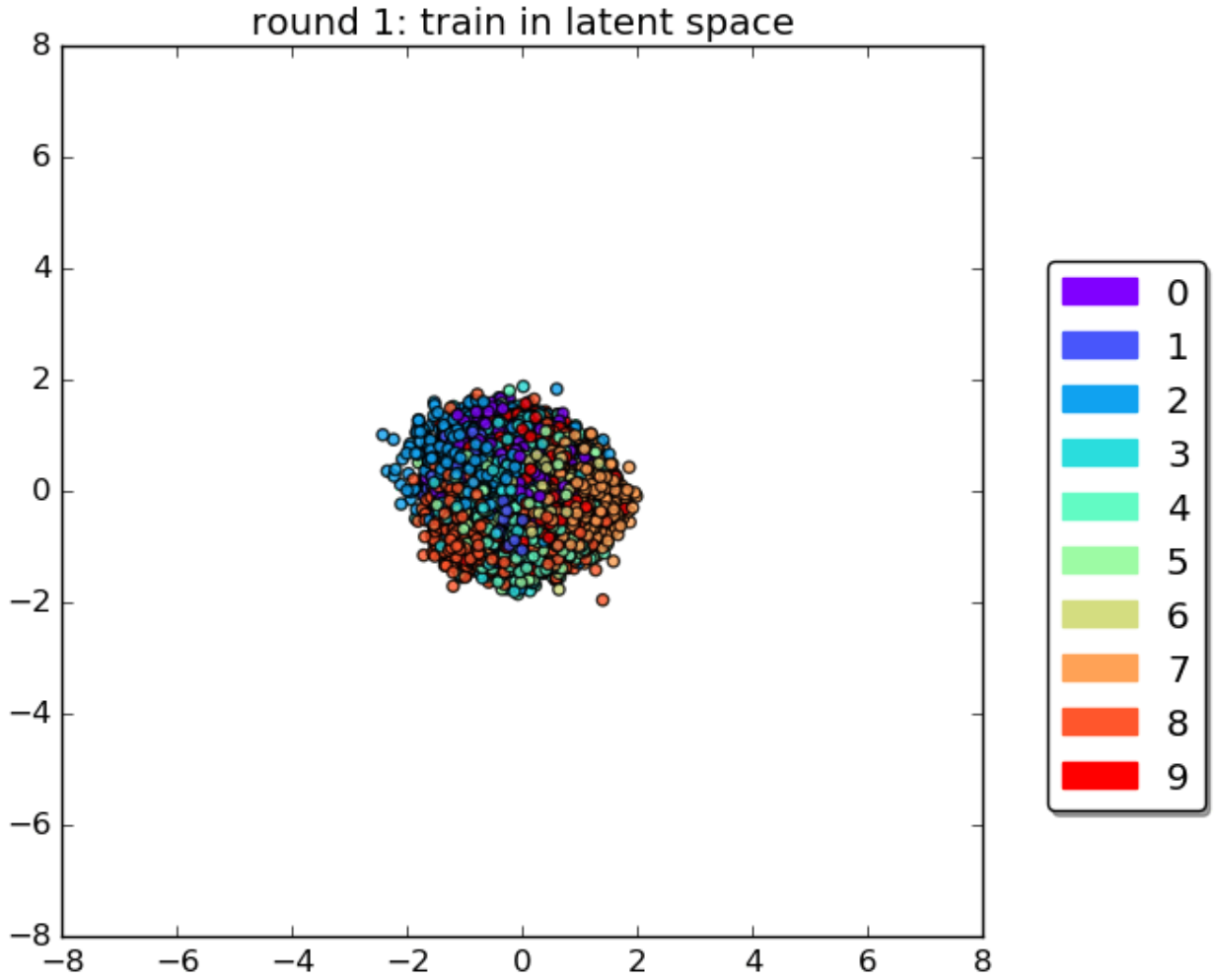
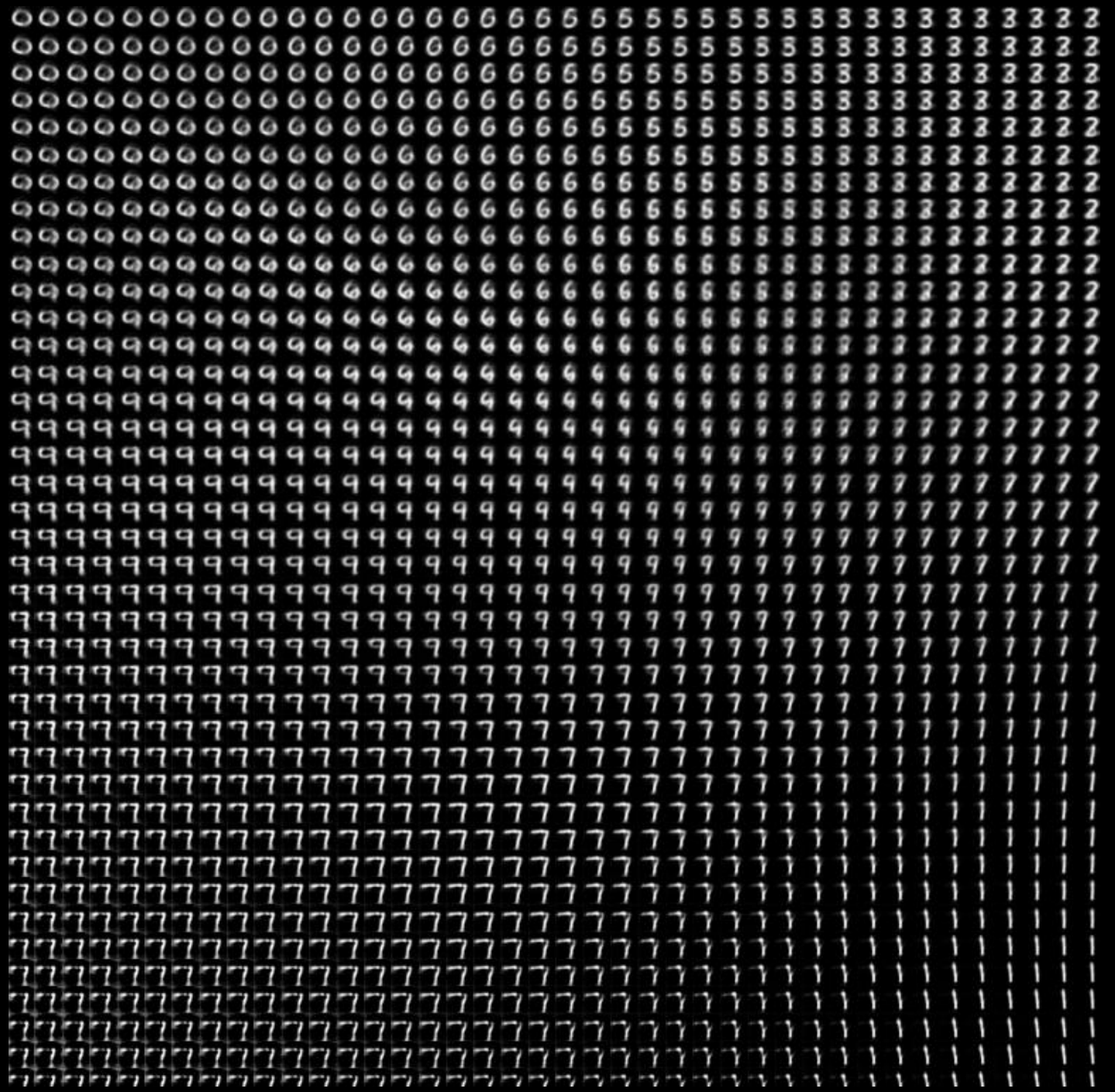


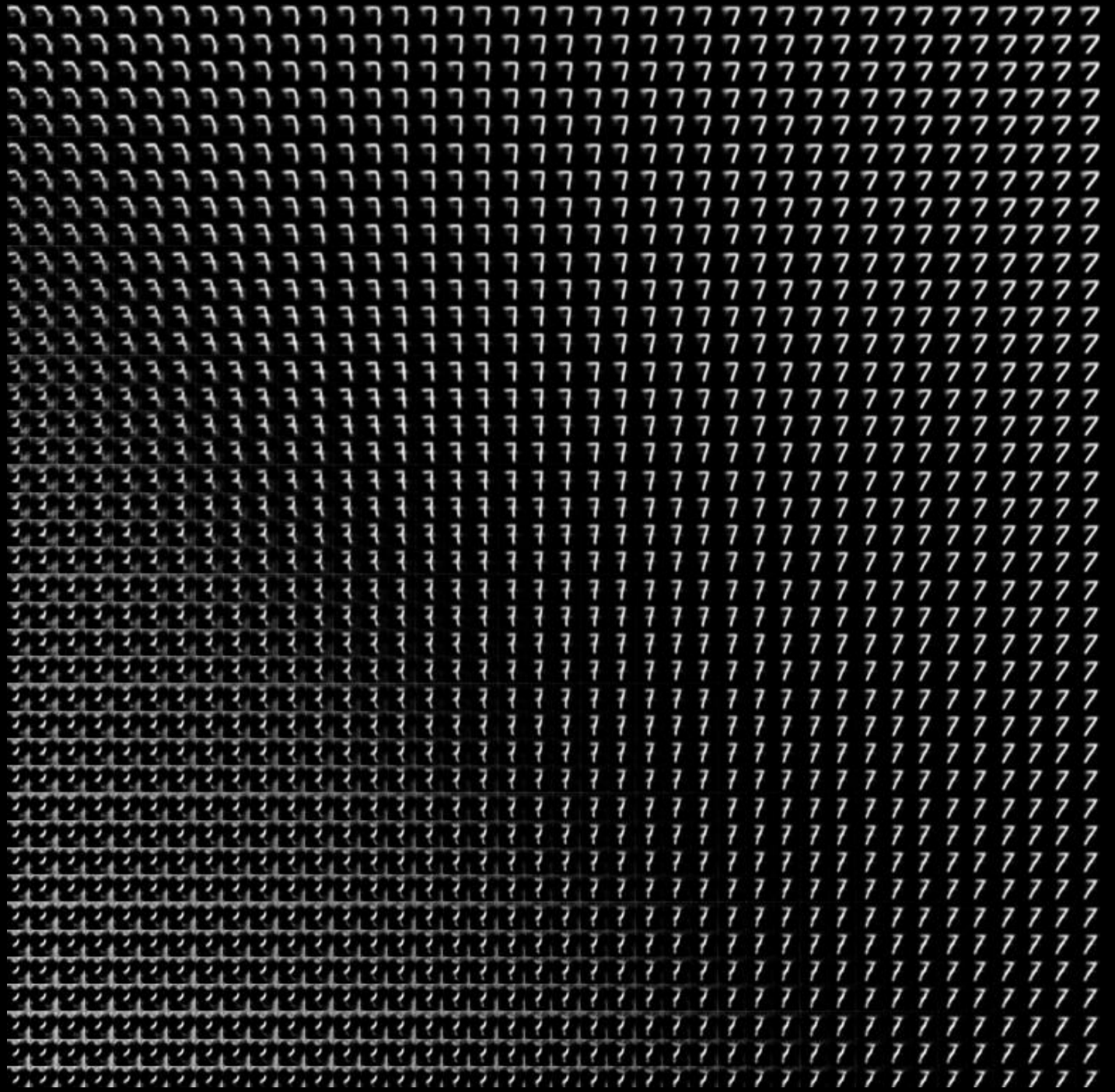
Image space → Latent space



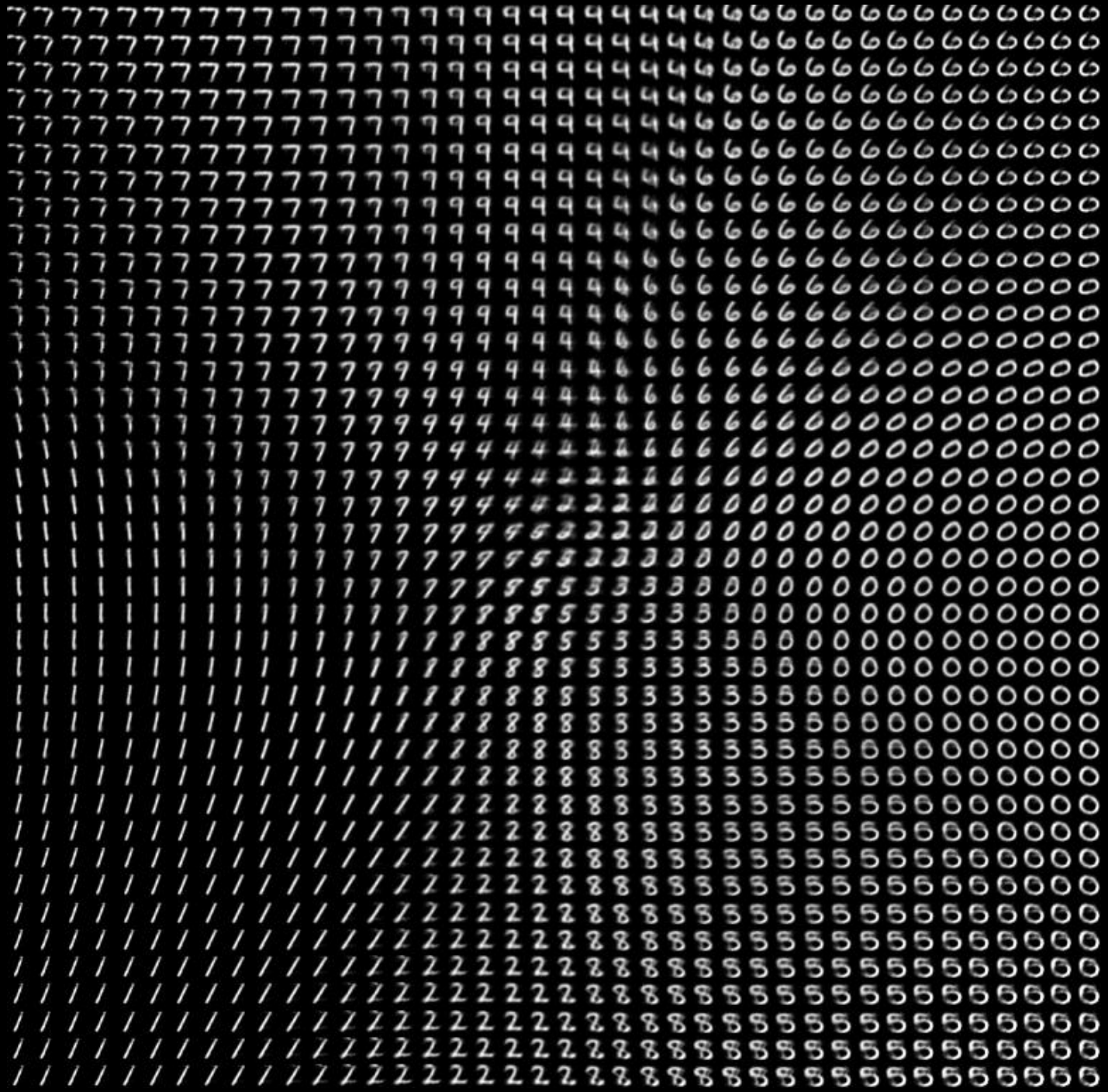
Autoencoder latent representation



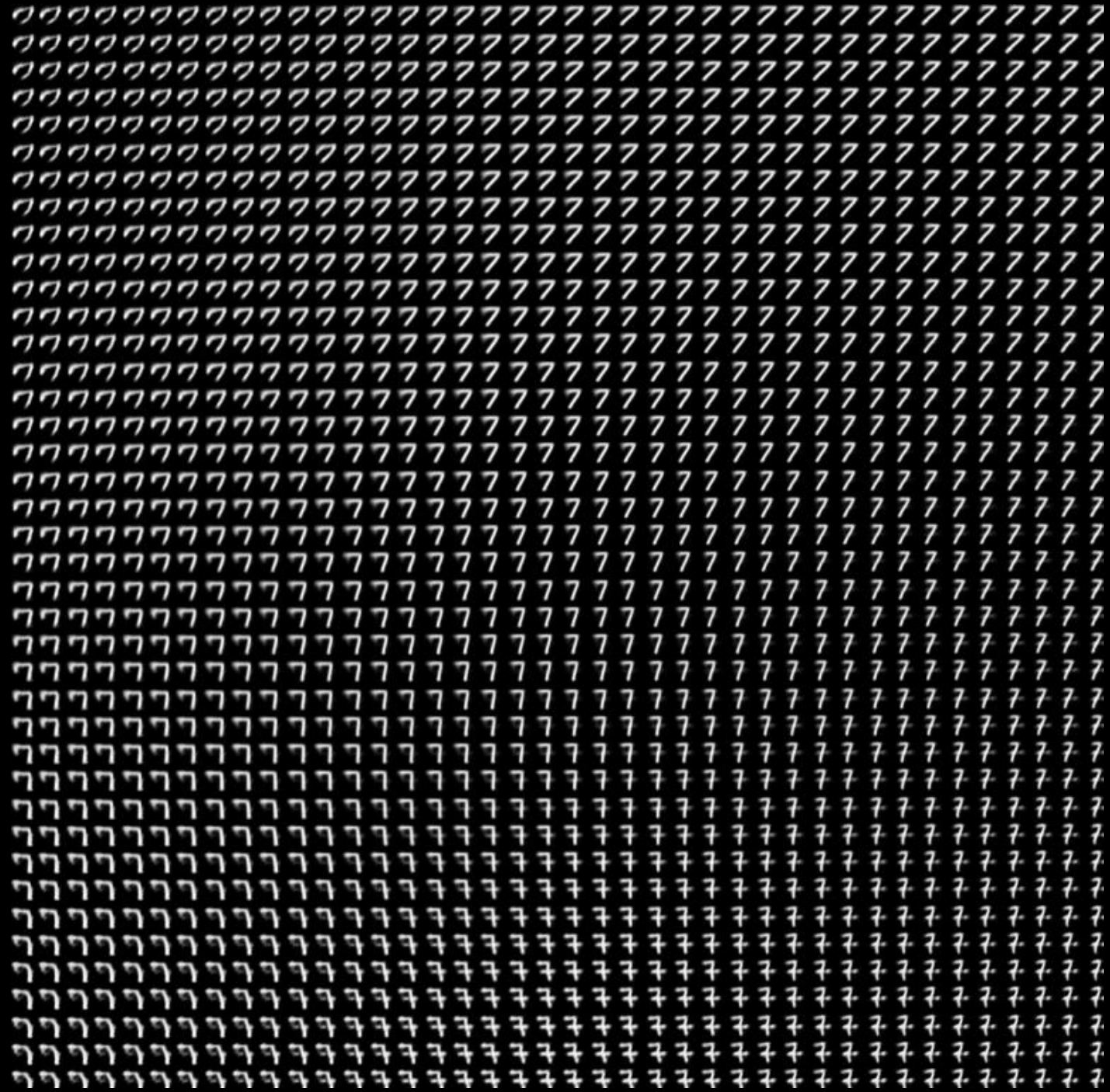
Autoencoder latent representation (digit 7)



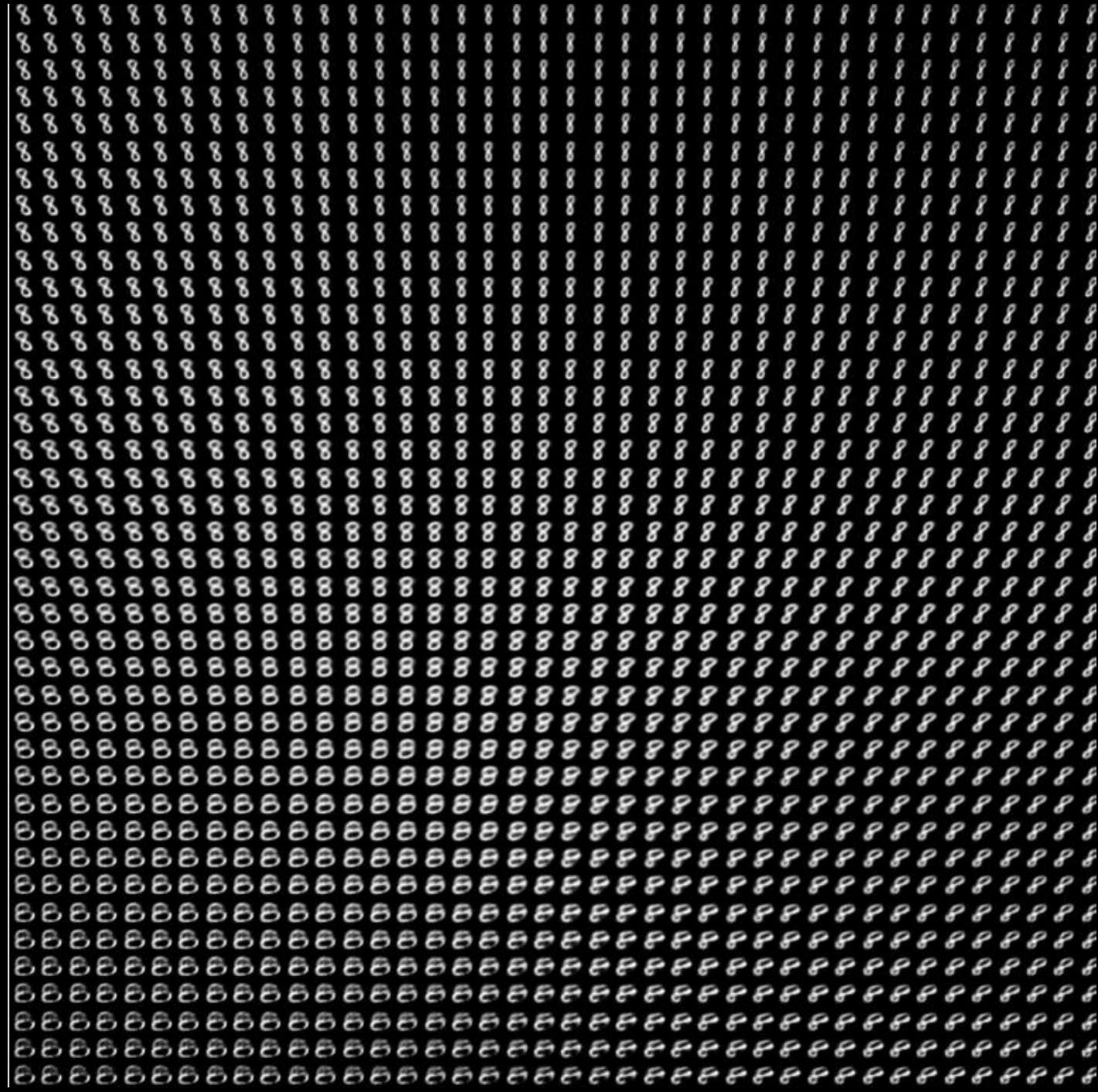
VAE latent representation



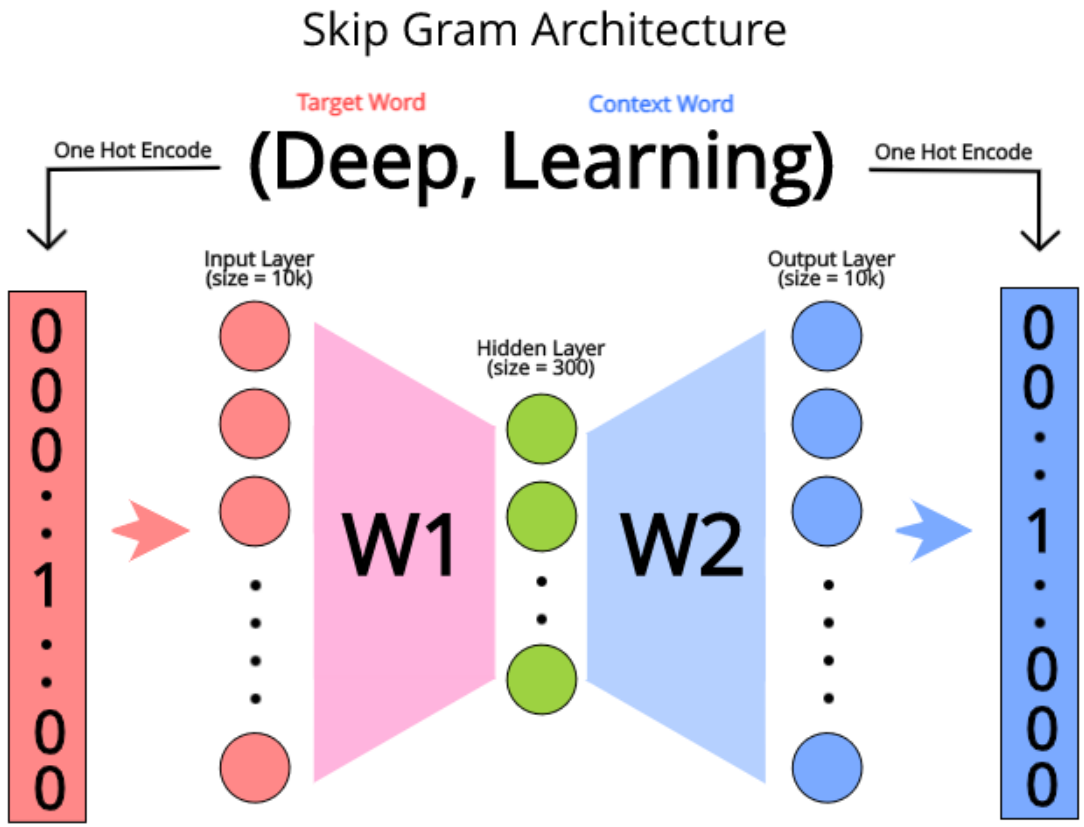
VAE latent representation (digit 7)

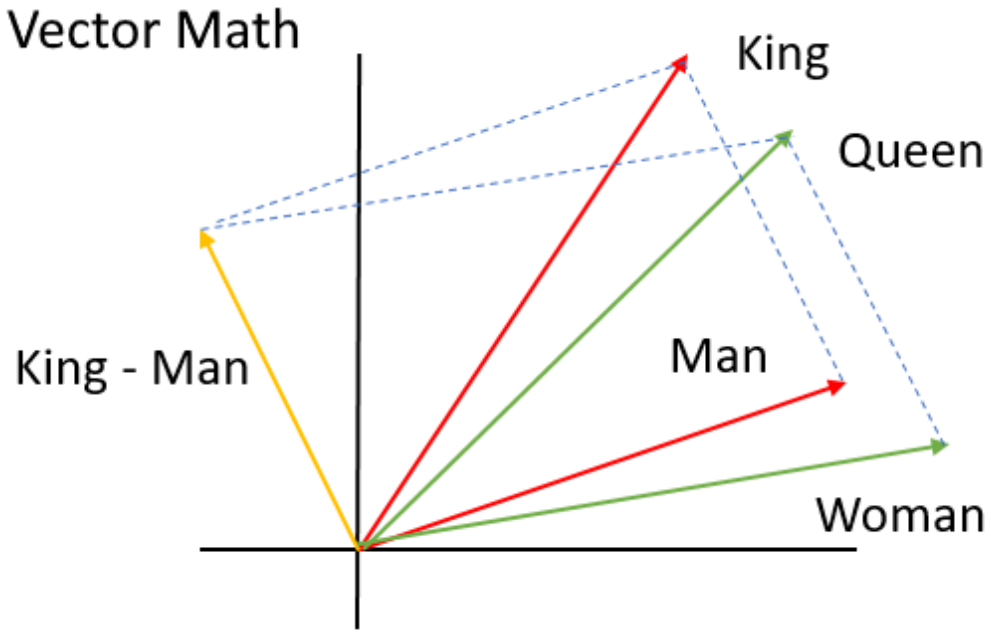


VAE latent representation (digit 8)



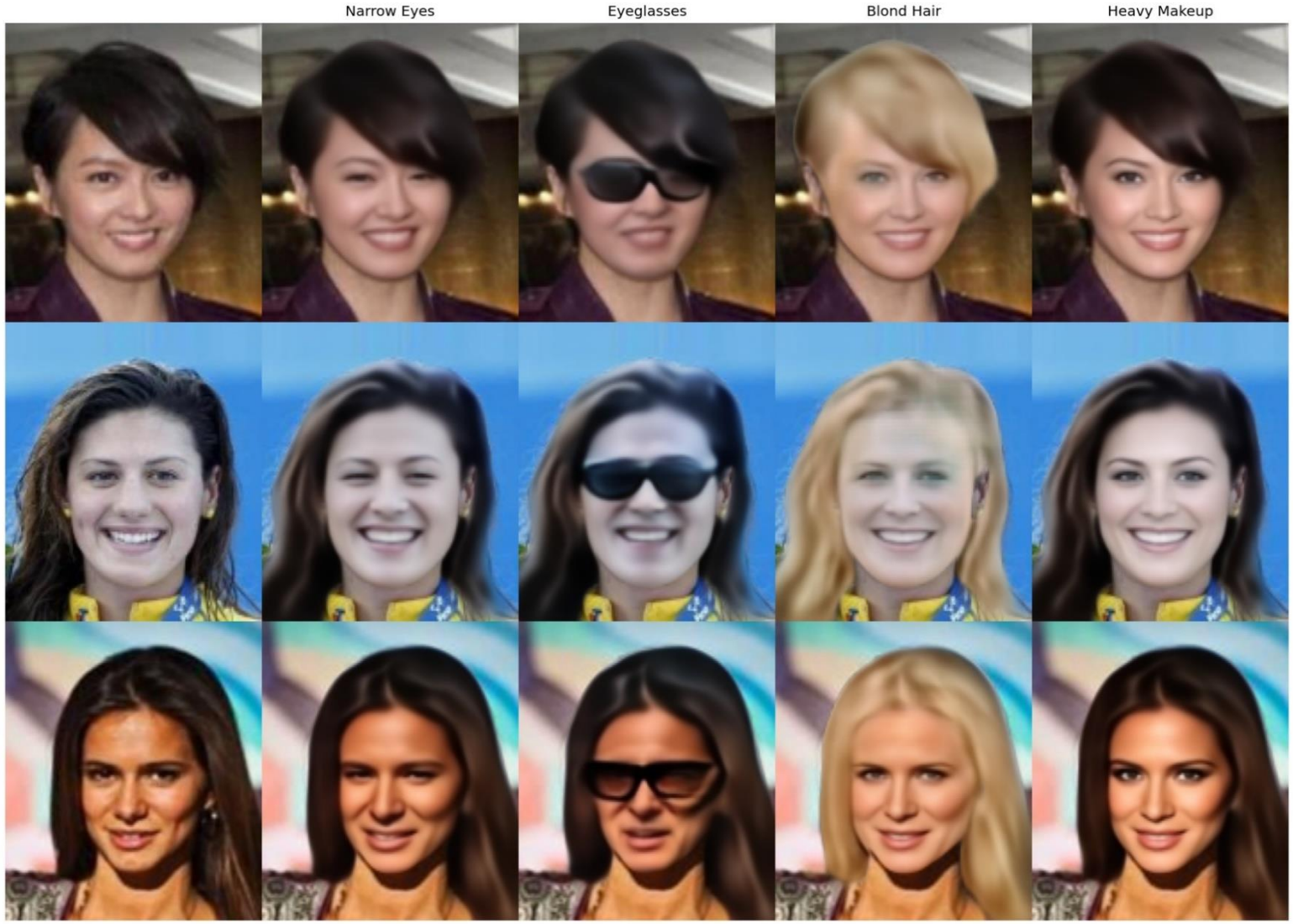
Word Vectors





https://medium.com/analytics-vidhya/word-embeddings-in-nlp-word2vec-glove-fasttext-24d4d4286a73

Changing Attributes

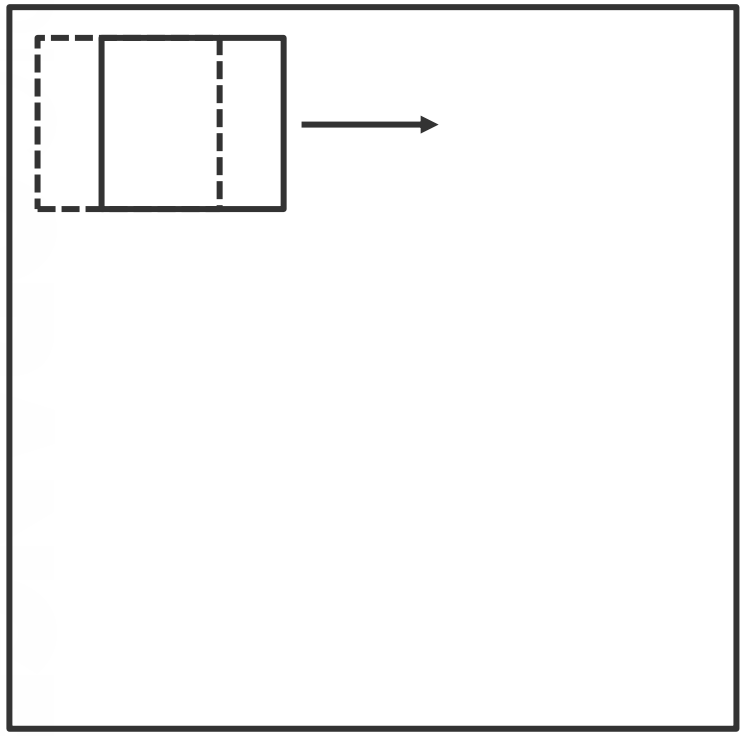


https://rtoledo.me/post/2021-05-31-edit-face-attributes-using-vae/edit-face-attributes-using-vae/

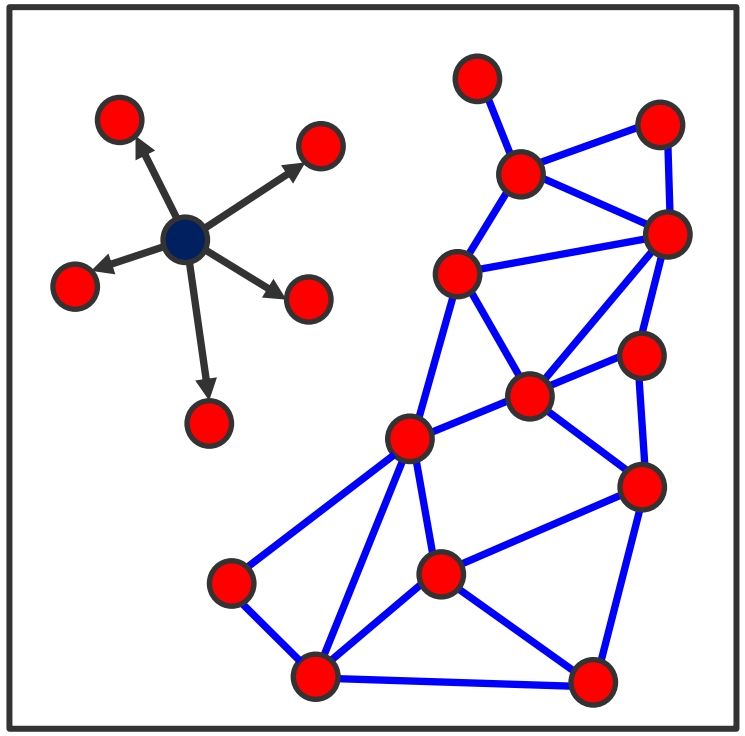
Describing the building blocks

- The classical physical descriptions (symmetry, etc) can be defined locally only in Bayesian sens
- We can argue that local descriptors are simple, if not necessarily known
- And the rules that guide their emergence are also simple, if not known

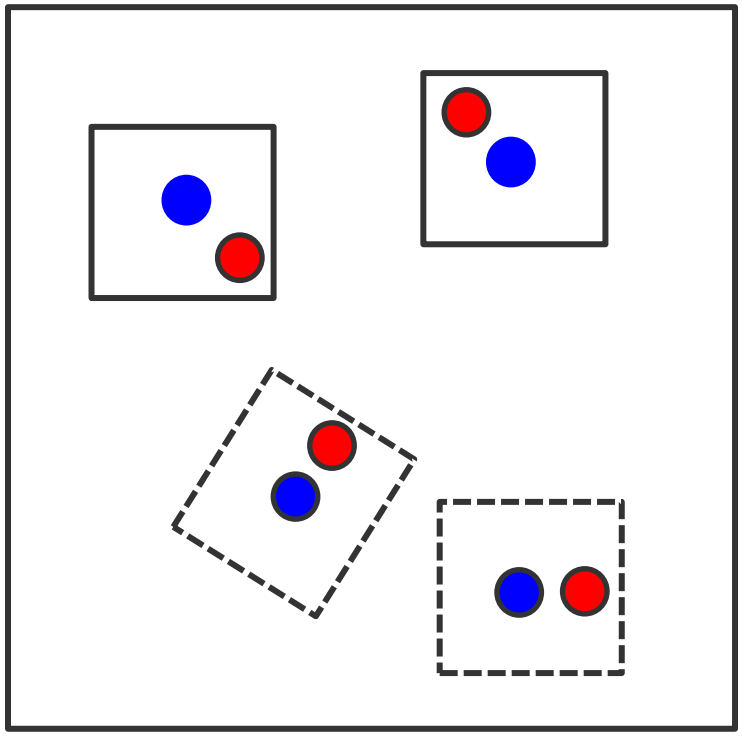
Continuous translational symmetry



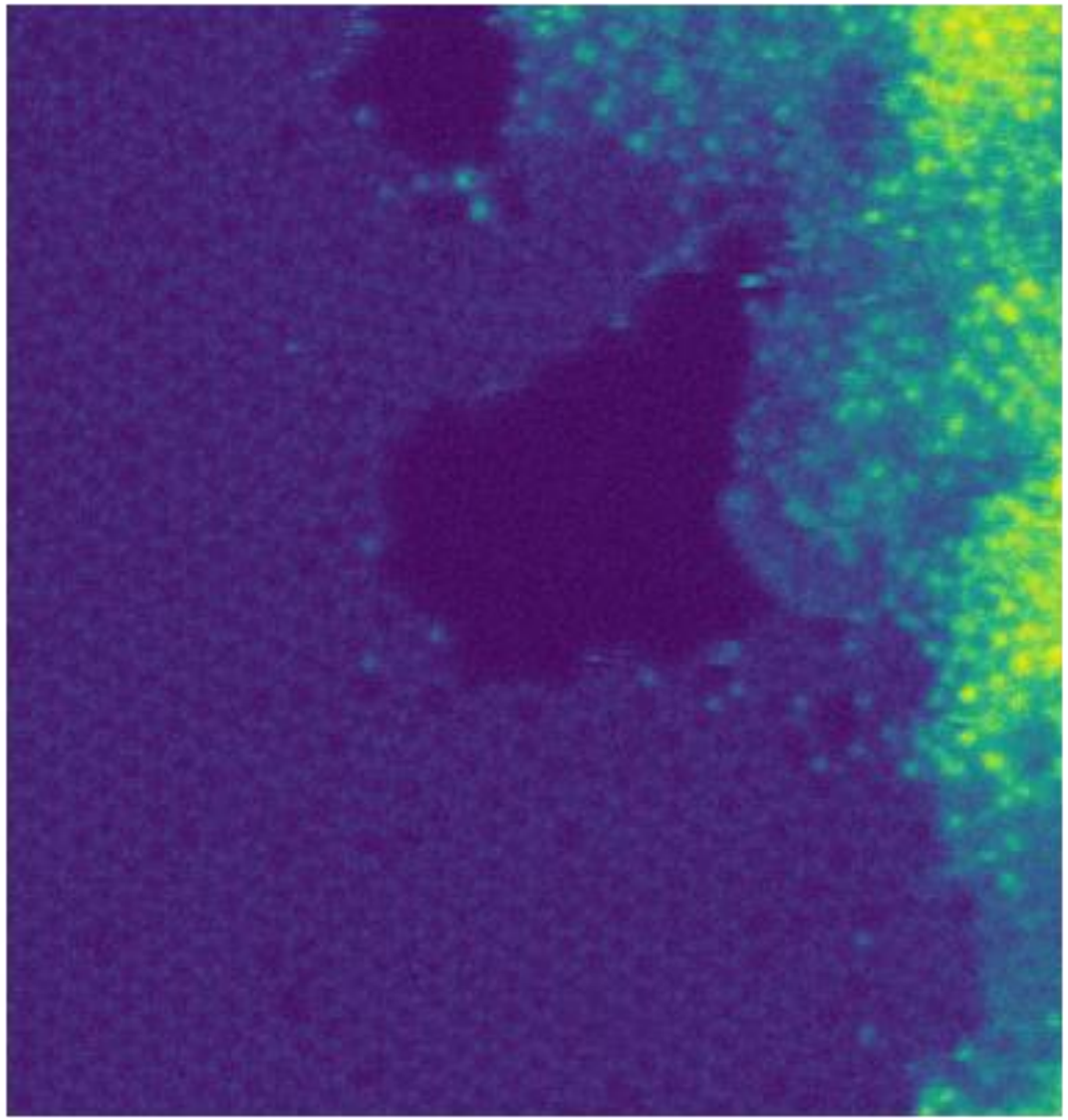
Atom based descriptions

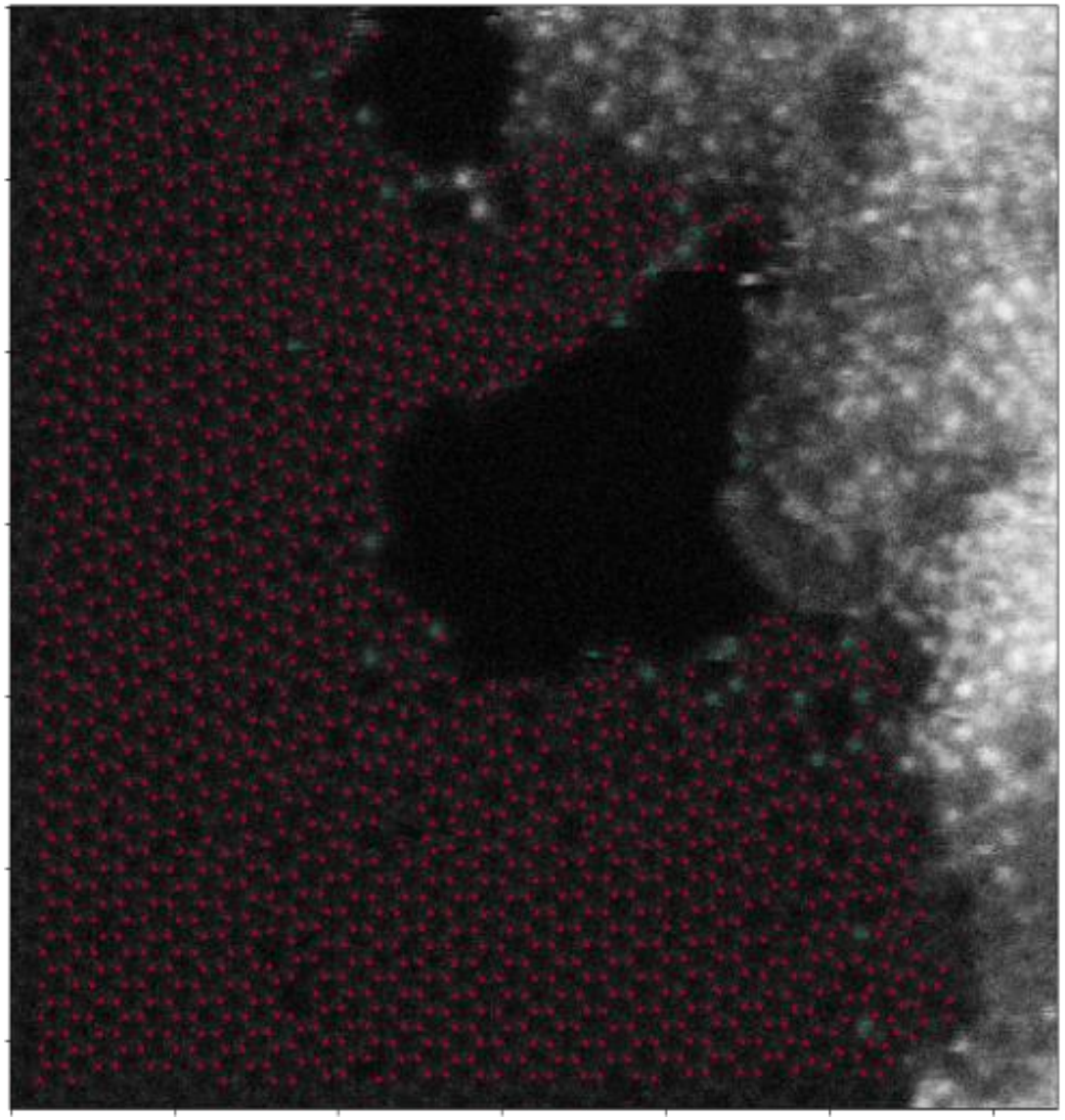


Localized sub-images

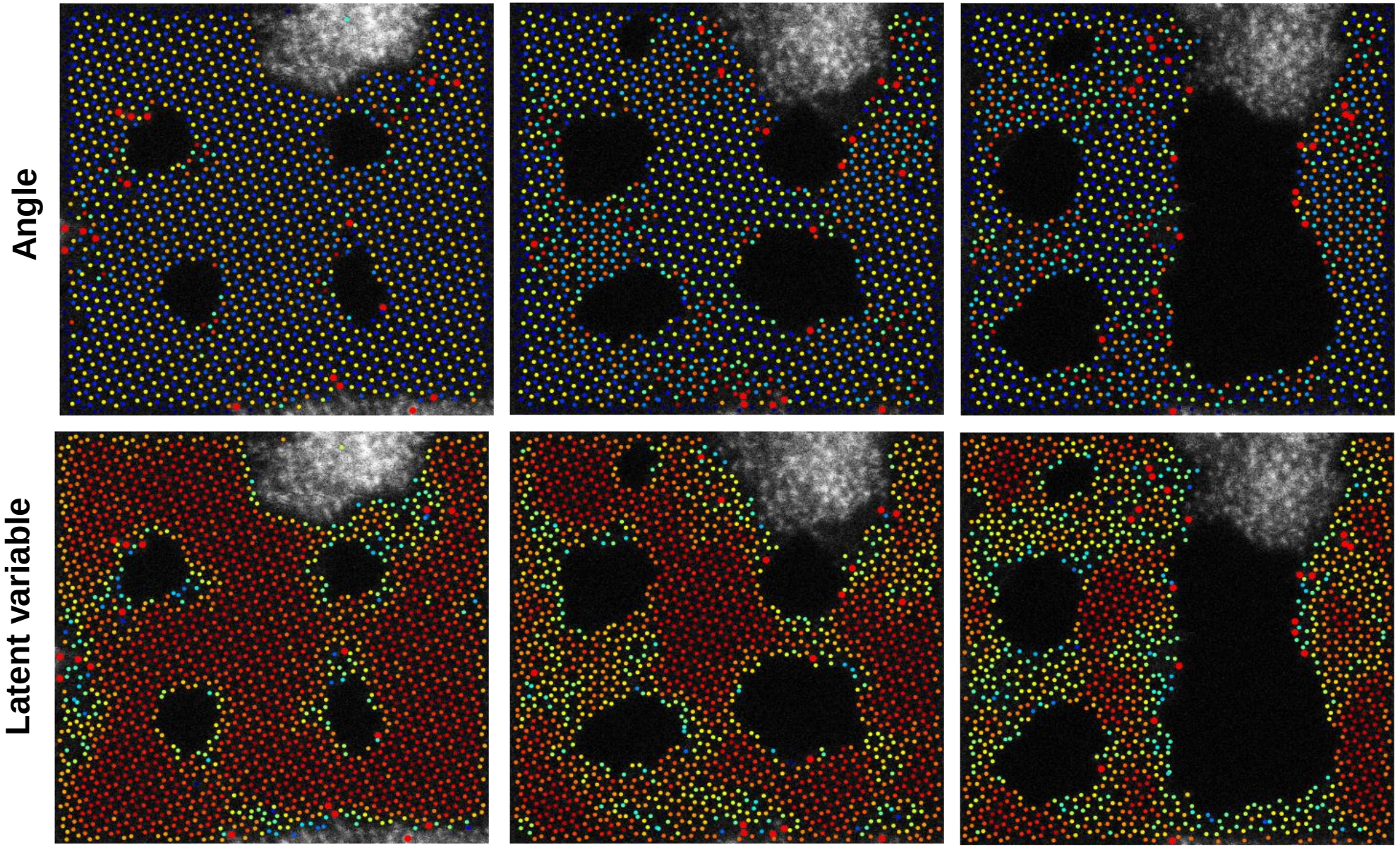


Off to chemically-disordered systems

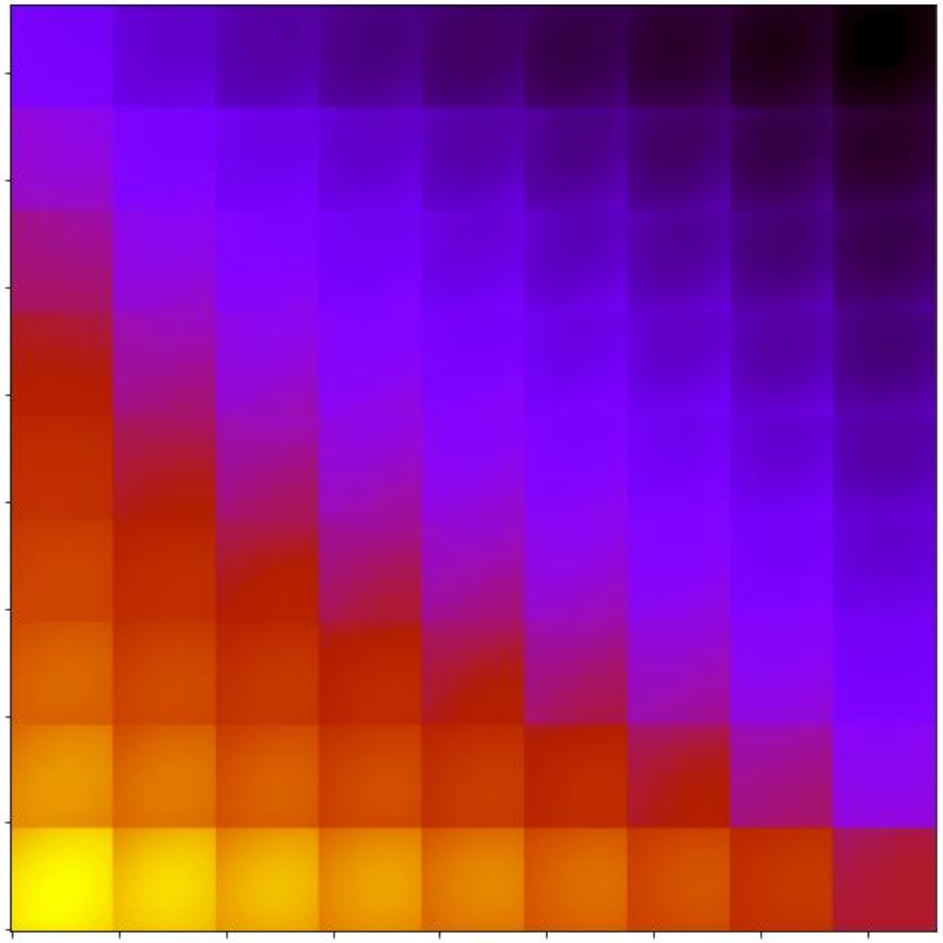


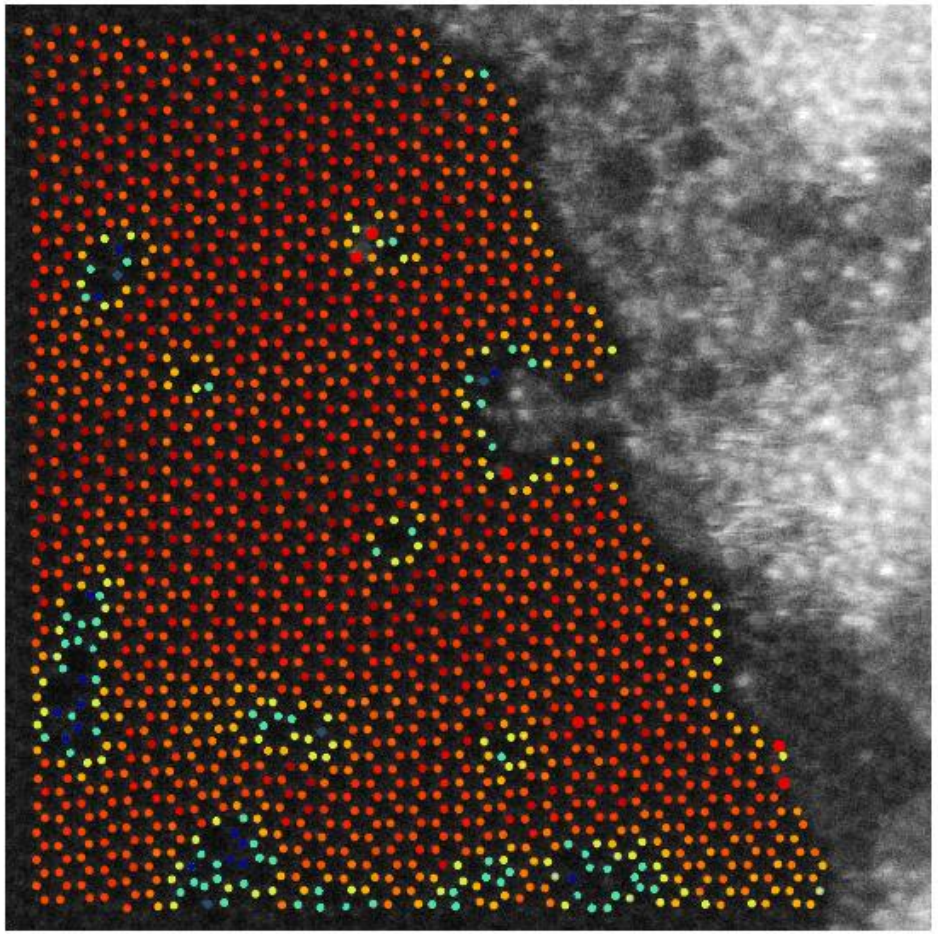


rVAE analysis at different time steps

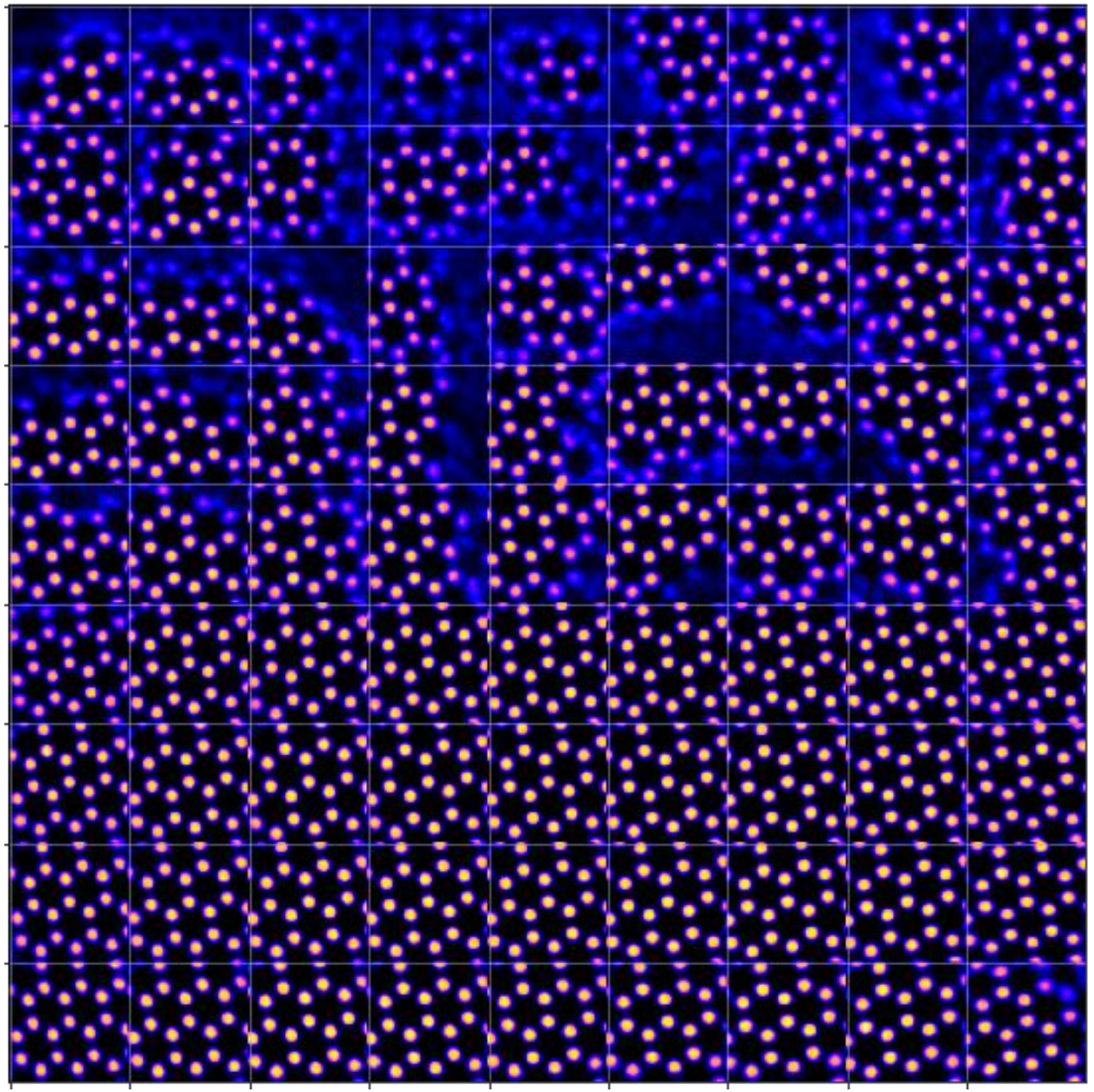


There is nothing as beautiful as training VAE





Next step: skip-rVAE

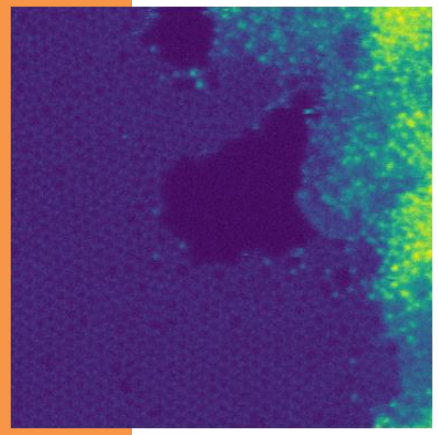


Unsupervised discovery of molecules

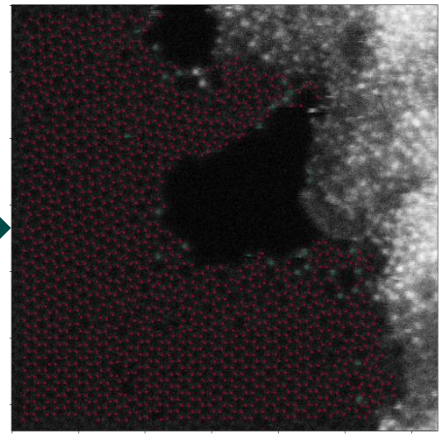
Atom exists, are discrete, and have known shape

Structures are parsimonious in nature

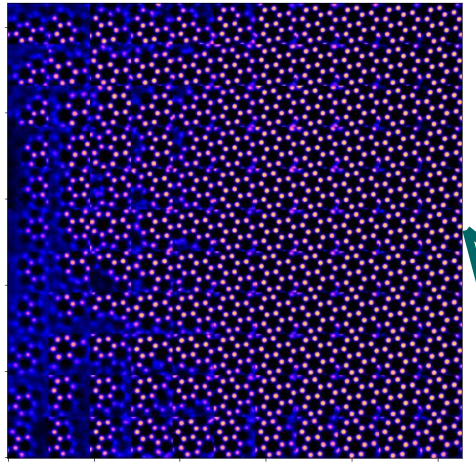
Raw observation



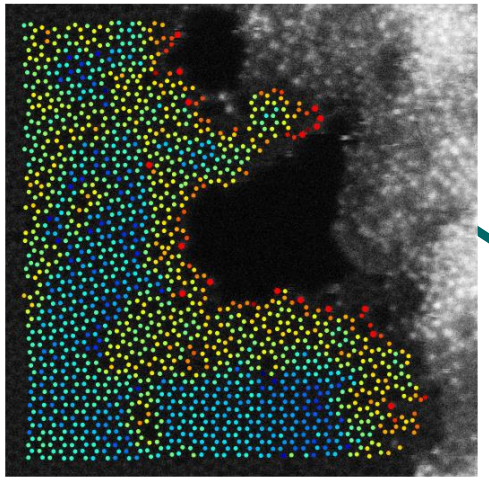
DCNN decoded



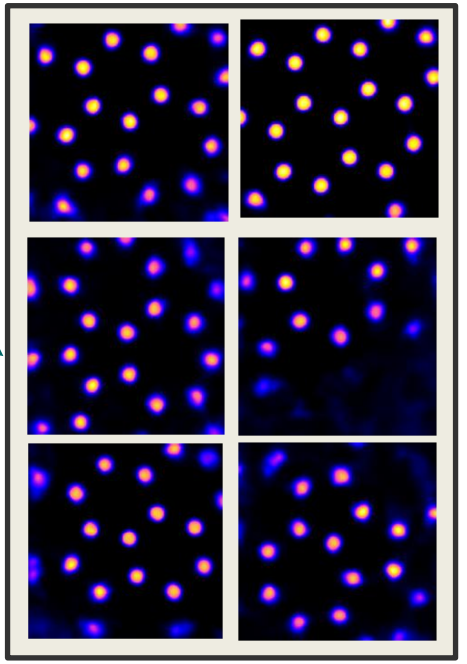
Latent space of srVAE



Latent labels



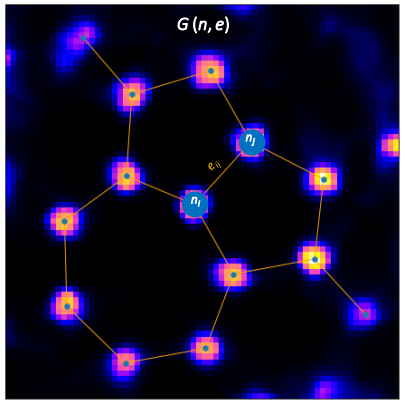
I. Molecules

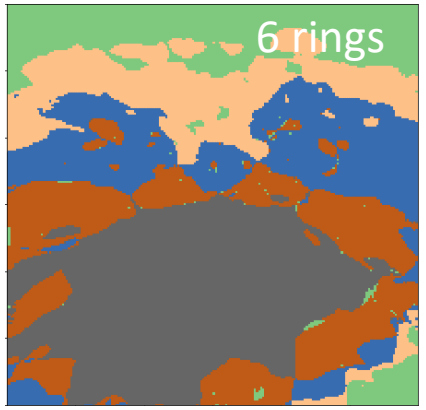


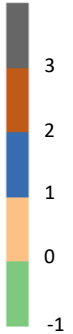
II. Reaction networks

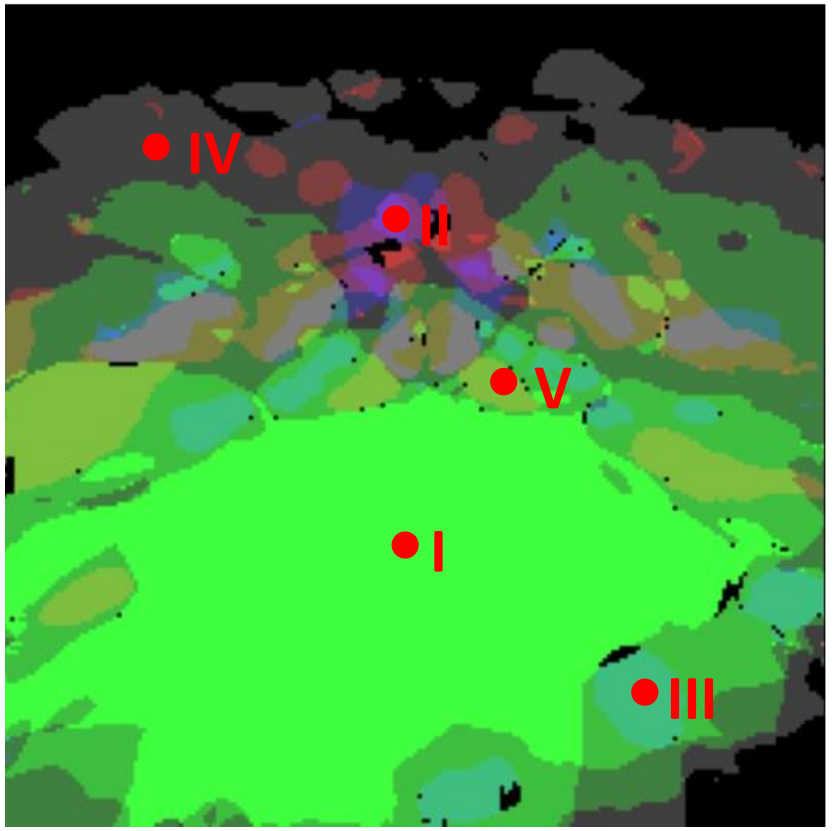
A -> B -> C

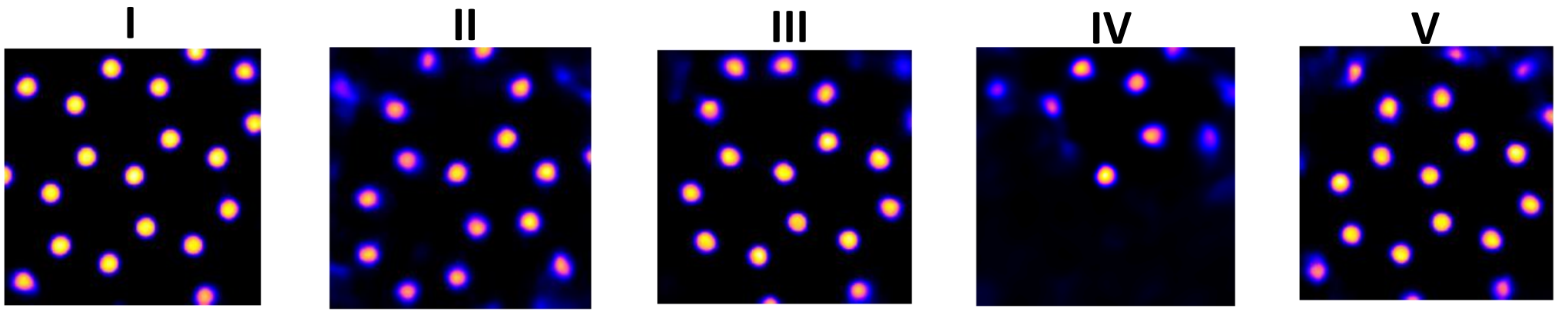
Exploring the latent space structure



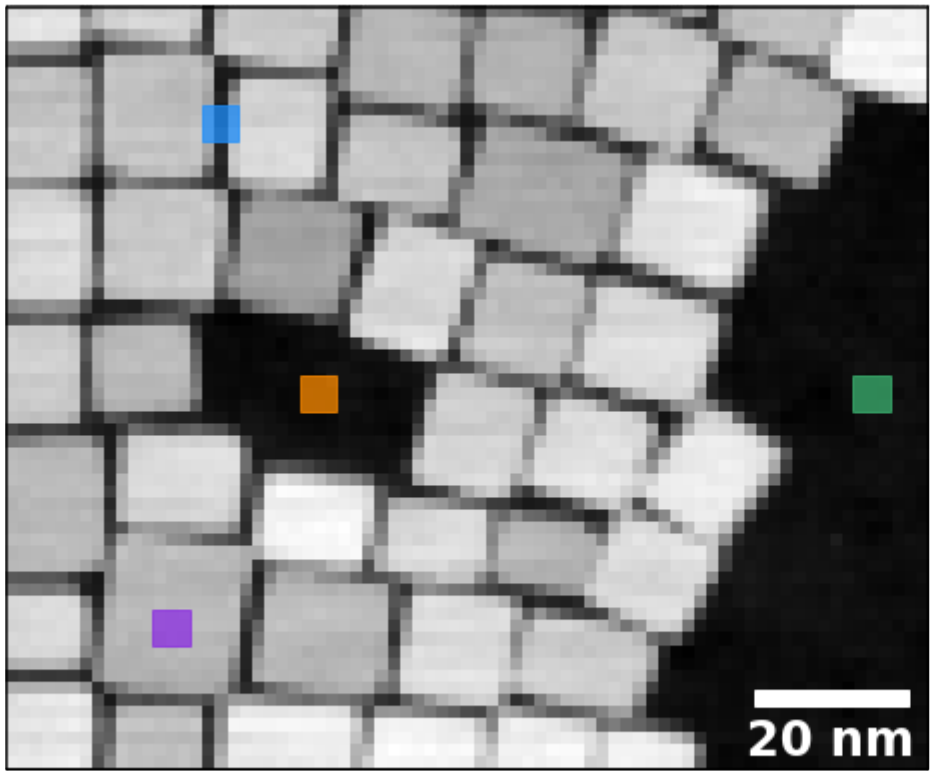


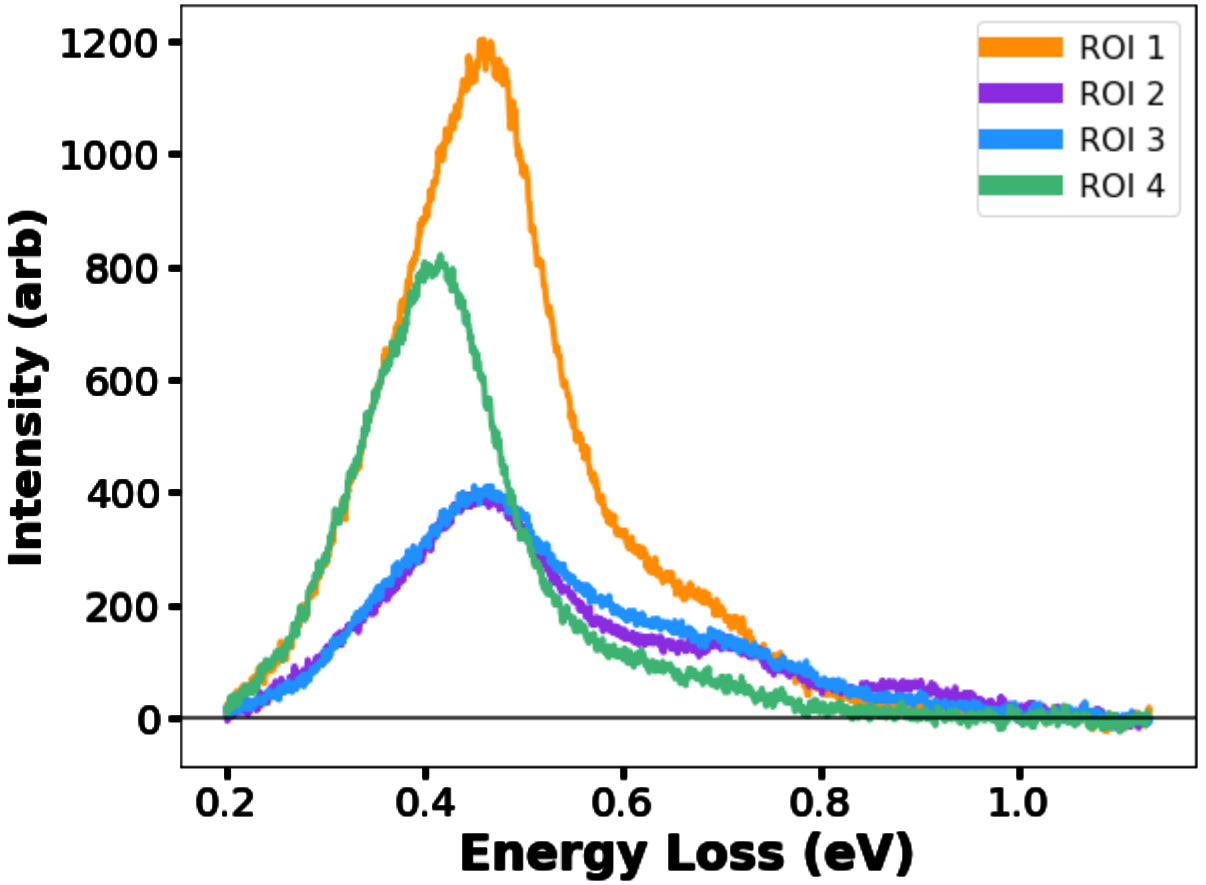






But what about the structure-property relations?

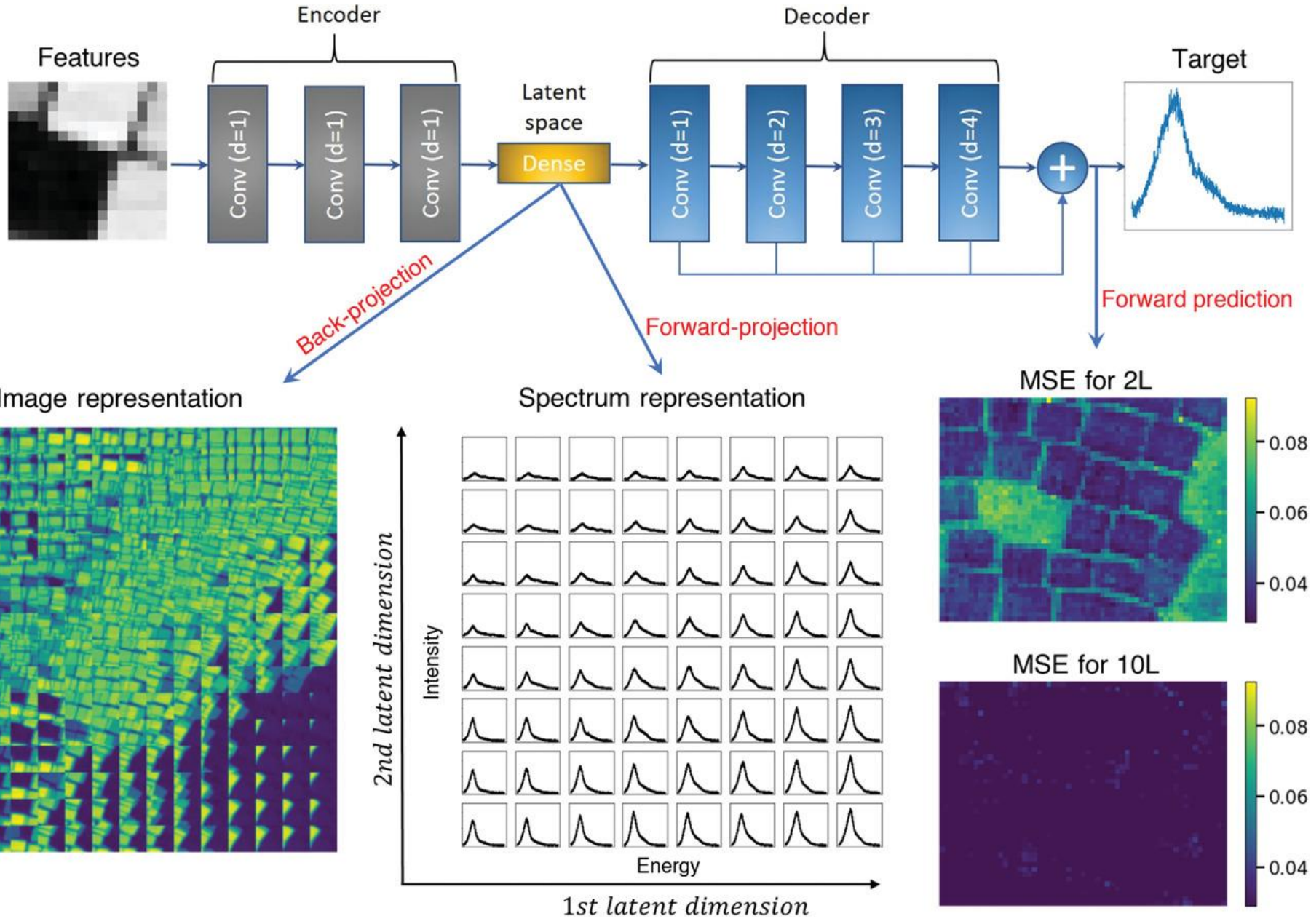




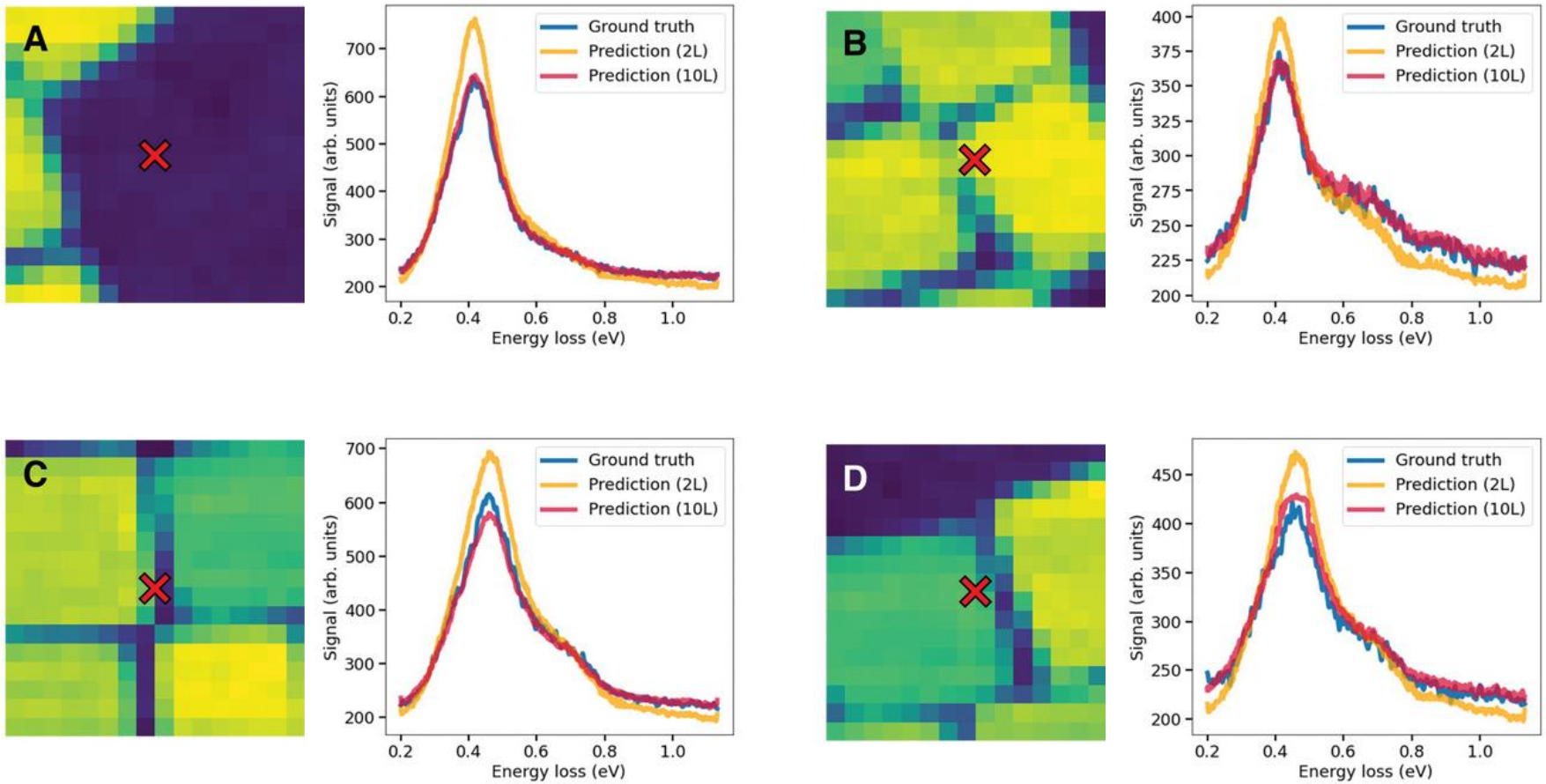
By inspection, we can note some characteristic aspects of spectra from specific types of geometries. However:

- How can we prove it and quantify this relationship?
- How universal is it for similar structures?
- Can we discover structures that will have the properties that we want?

Im2spec: latent space visualization

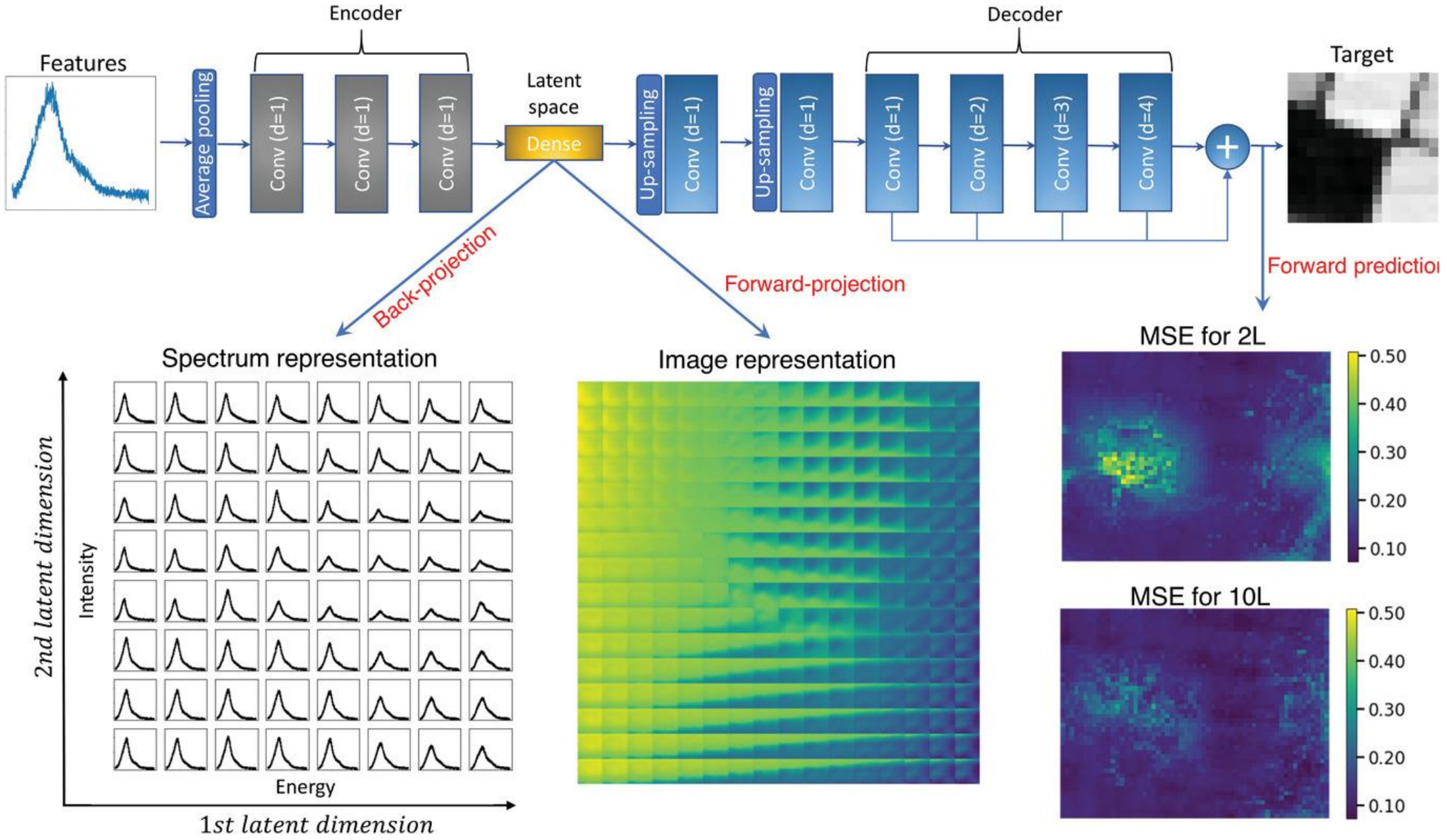


Im2spec prediction

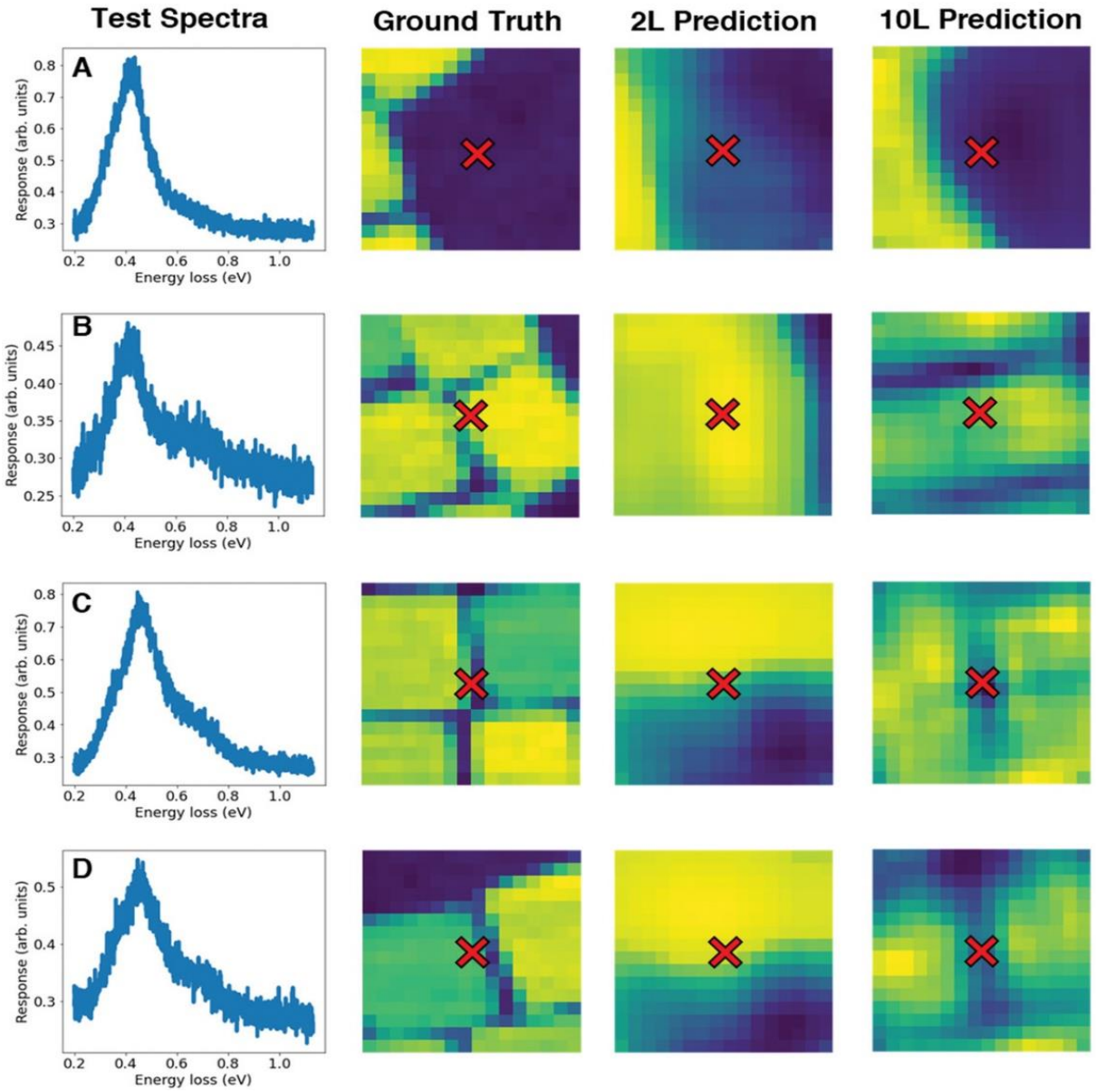


- After training, predict the spectral response of a geometric arrangement that the network has **never encountered**
- **Library** of geometric-plasmonic relationships
- Can be used for solution of inverse design in nanophotonics and other fields

Spec2im


Encoder
Decoder
Features
Average pooling
Conv (d=1)
Conv (d=1)
Conv (d=1)
Latent space
Dense
Up-sampling
Conv (d=1)
Up-sampling
Conv (d=1)
Conv (d=1)
Conv (d=2)
Conv (d=3)
Conv (d=4)
Target
Back-projection
Forward-projection
Forward predictio
Spectrum representation
Image representation
MSE for 2L
MSE for 10L
2nd latent dimension
Intensity
Energy
1st latent dimension
0.50
0.40
0.30
0.20
0.10

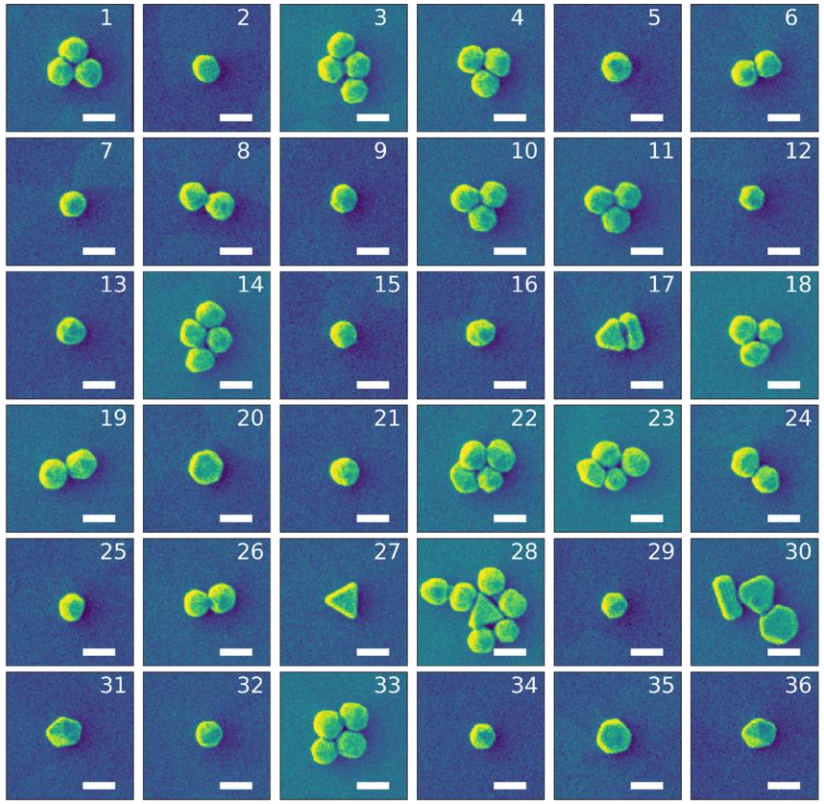
Spec2im predictions



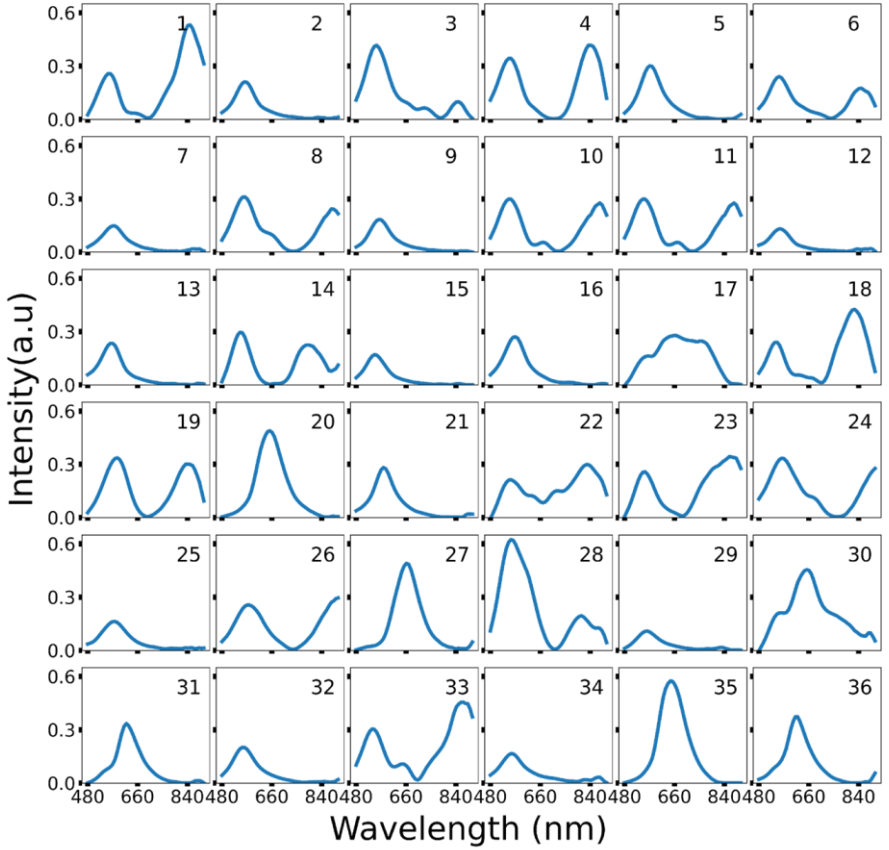
- After training, predict the spectral response of a geometric arrangement that the network has **never encountered**
- **2L** and **10L** refers to number of latent dimensions chosen

Dual VAE: structure-property relationships

SEM images: “Structure Information”

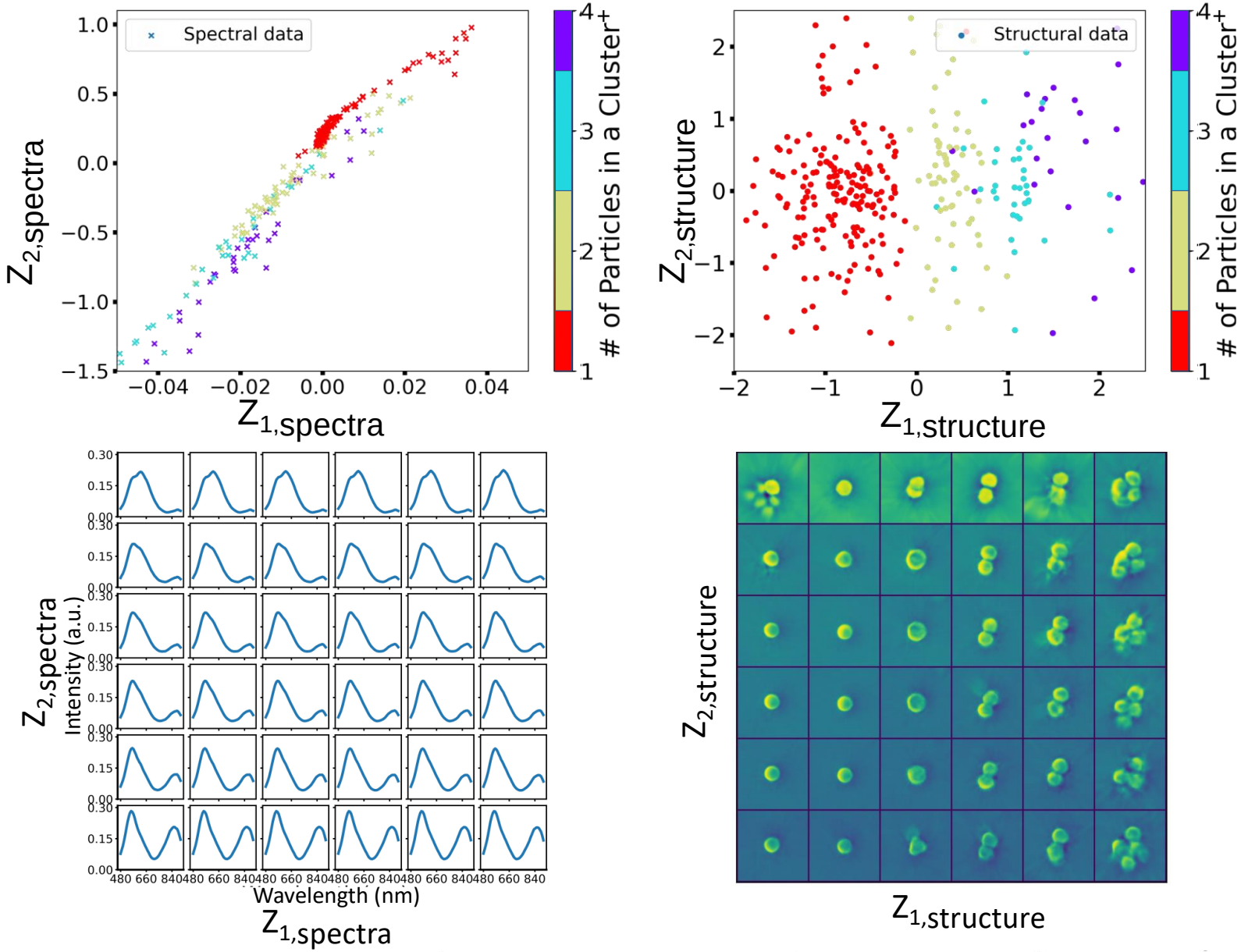


Hyperspectral microscope: “Property Information”

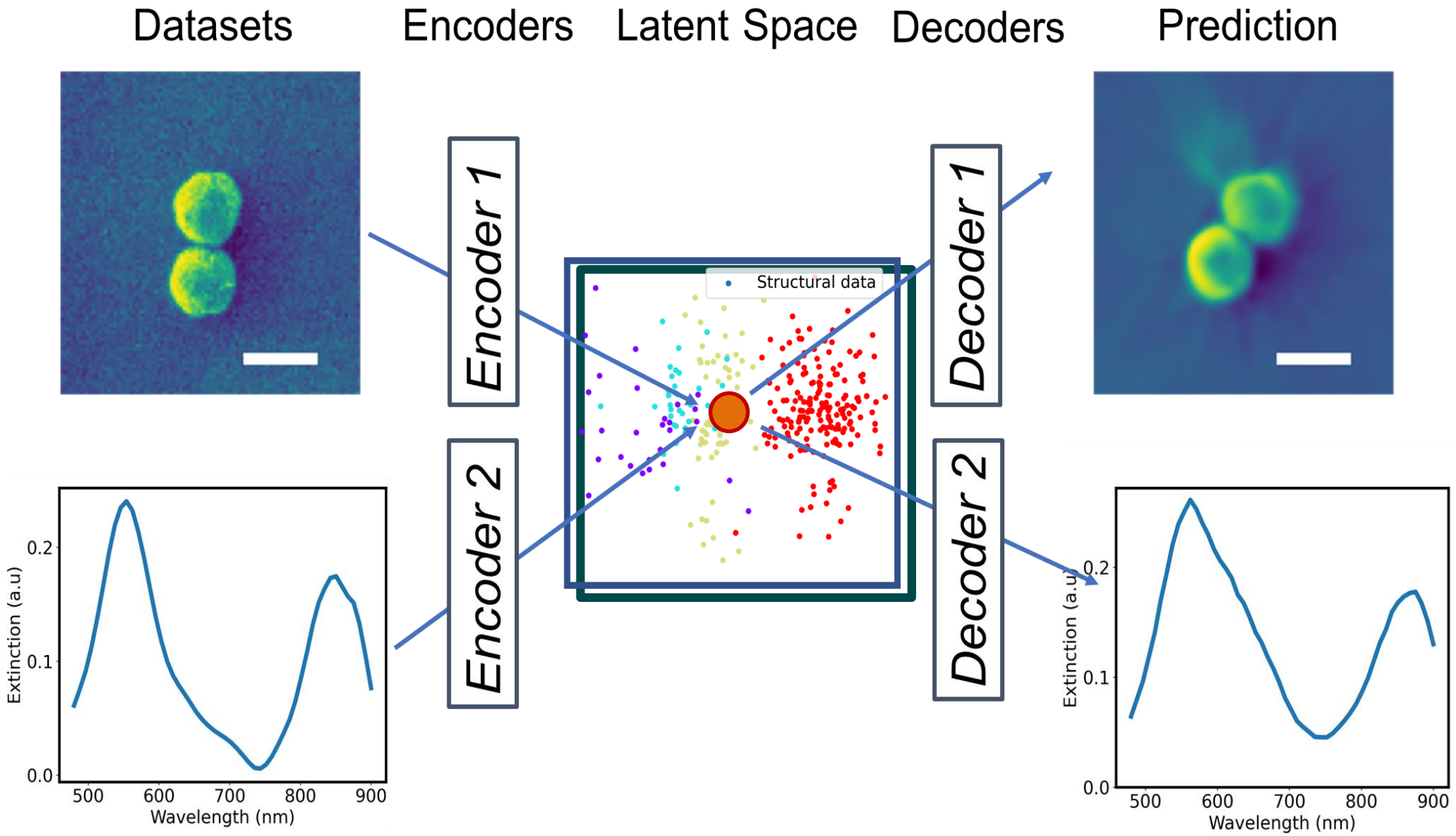


- Far field optical spectroscopy: images and spectra
- Here, we also have simple labels (number of clusters)

Separated VAE

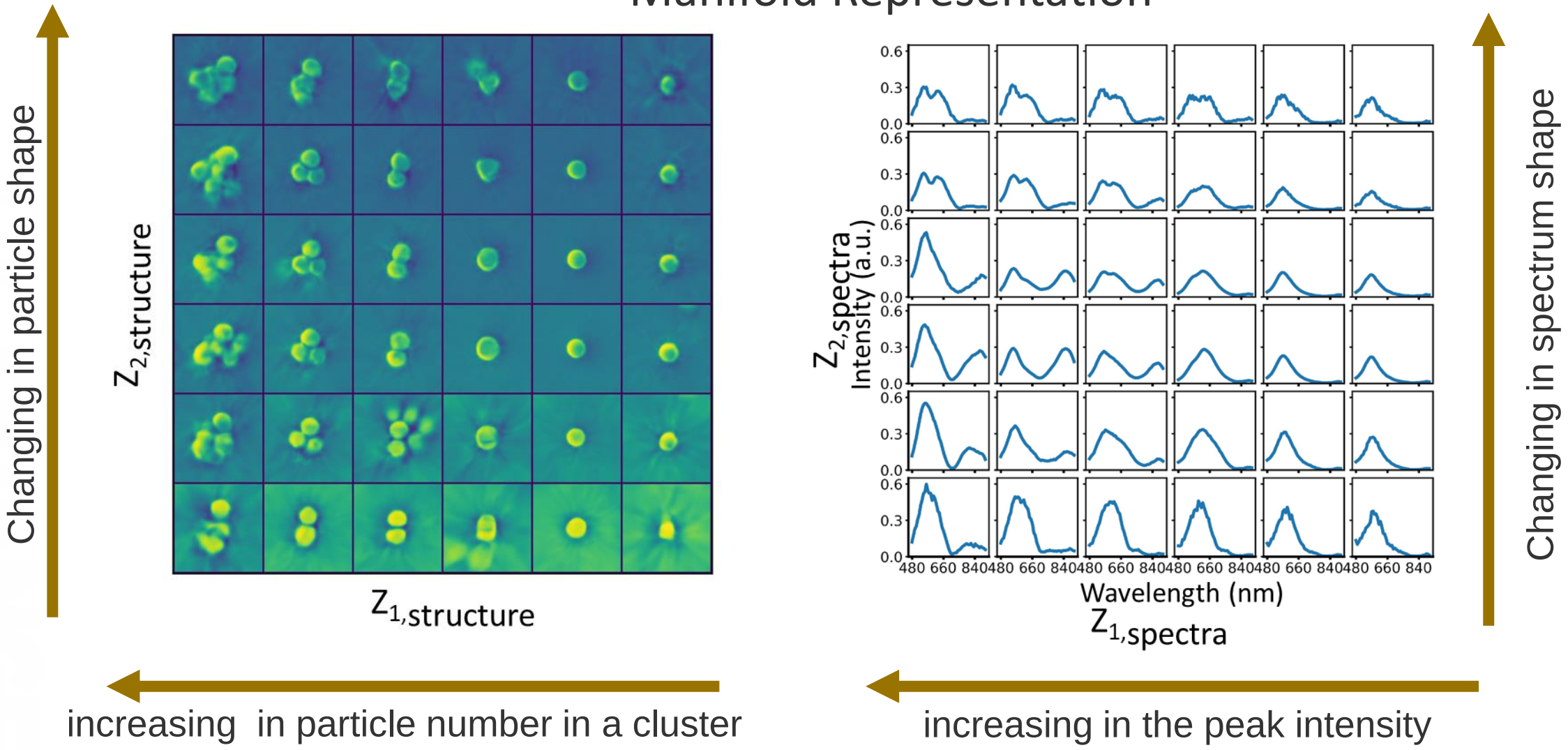


Dual VAE


Datasets
Encoders
Latent Space
Decoders
Prediction
Encoder 1
Encoder 2
Structural data
Decoder 1
Decoder 2
Extinction (a.u)
0.0
0.1
0.2
500
600
700
800
900
Wavelength (nm)

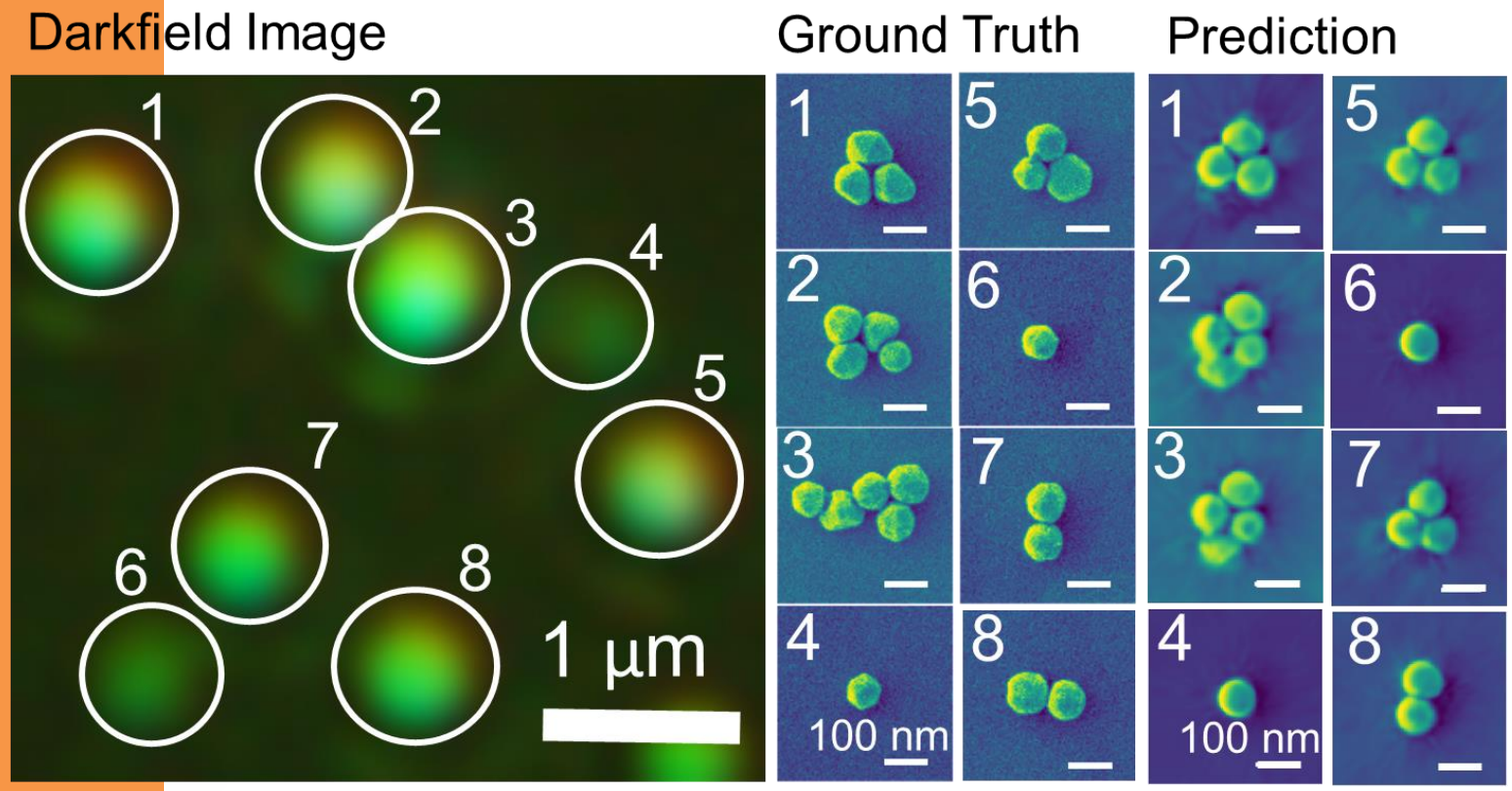
Dual VAE: Latent Representations

Manifold Representation

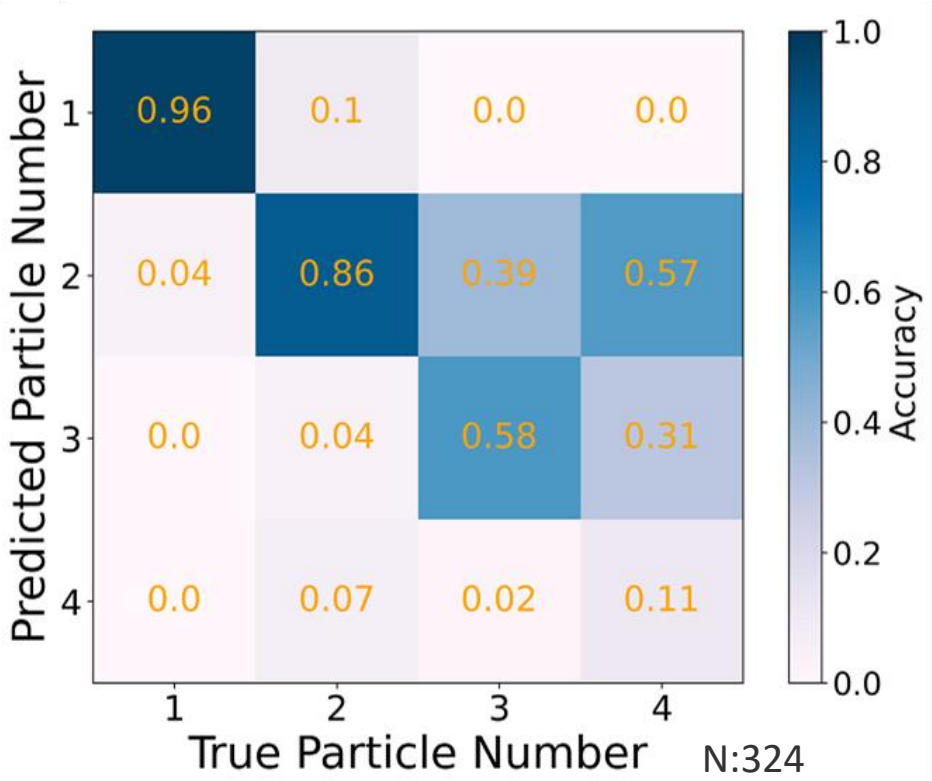


Dual VAE: Predictions

Example



Overall Particles



N:324

Questions to ask when using ML

Data science:

- What is the dimensionality of feature and target spaces
- How variable are the features in these spaces (VAE can help)
- How much data do I have?

Domain knowledge:

- How predictive do I expect this relationship to be
- What other factors matter from materials side
- Can measurements introduce additional factors of variance?

Experiment planning:

- How much data can I have?
- Is it a static learning problem, or am I interested in ML-assisted experiment?