Data Science and AI for Medicine Training School

TRAINING: Introduction to Deep Learning

SPEAKER: Michaela Unger

GEFÖRDERT VOM
Bundesministerium
für Forschung, Technologie
und Raumfahrt

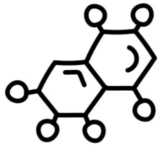
SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Deep Learning is everywhere

Deep Learning in in Medicine

Medical Imaging,
e.g. Radiology, Pathology, Dermatology

Drug Discovery

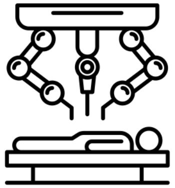


Disease Management,
e.g. Chronic Disease, Mental Disease

Diagnosis and Triage

Application Areas

Robotics and Surgery

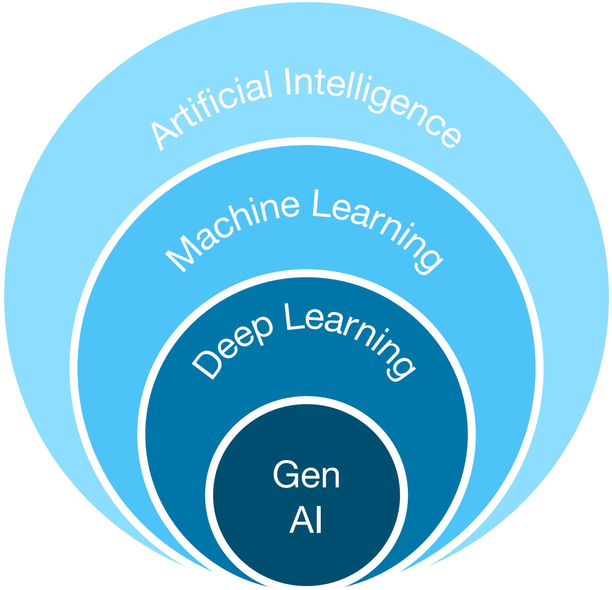


Decision Support / Personalized Medicine

Data Structuring

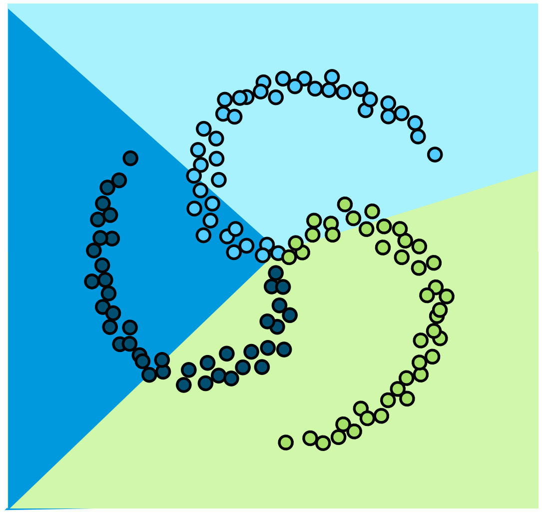
Virtual Health Assistants

Why Machine Learning is not enough sometimes…

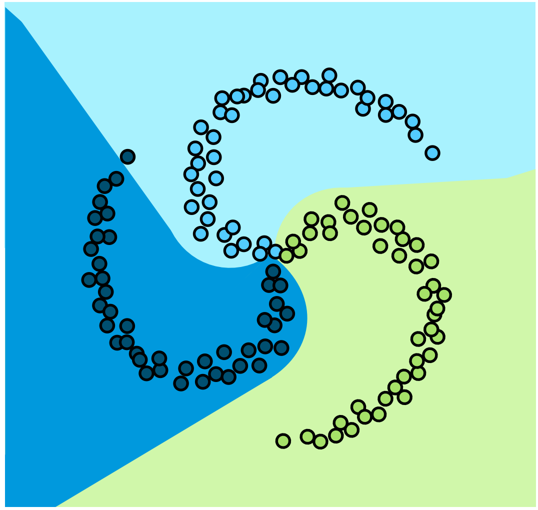


	Feature 1	Feature 2	Feature 3
Pat. 1	21	7	8
Pat. 2	5	35	9
Pat. 3	87	58	3

- Raw/ unstructured data
- Unknown features
- Manual labour
- Complex interactions



Deep learning can be used to model non-linear relations



History of DL

1943
McCulloch & Pitts first artificial neuron

1958
Rosenblatt's perceptron

1969
Minsky & Papert "Perceptron" book

Deep Learning Winter

1980
Fukishima's Neocognition

1986
Rumelhart, Hinton, Williams invent backpropagation algorithm

1989
LeCun engineers Convolutional Neural Networks

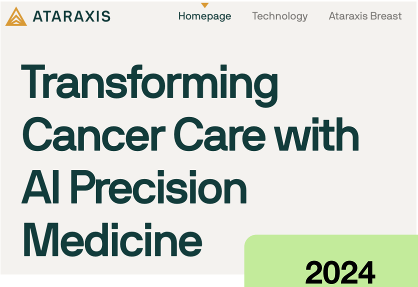
Deep Learning revival

2009
ImageNet

2012
GPU acceleration

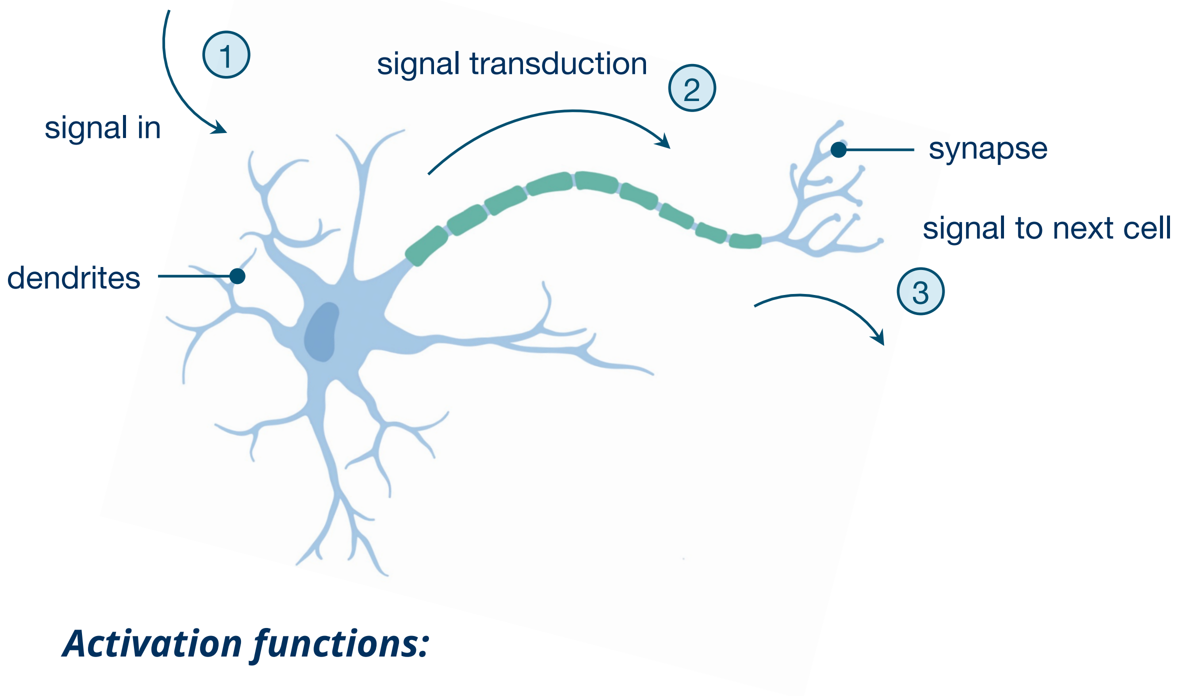
2017
Transformer Neural Networks

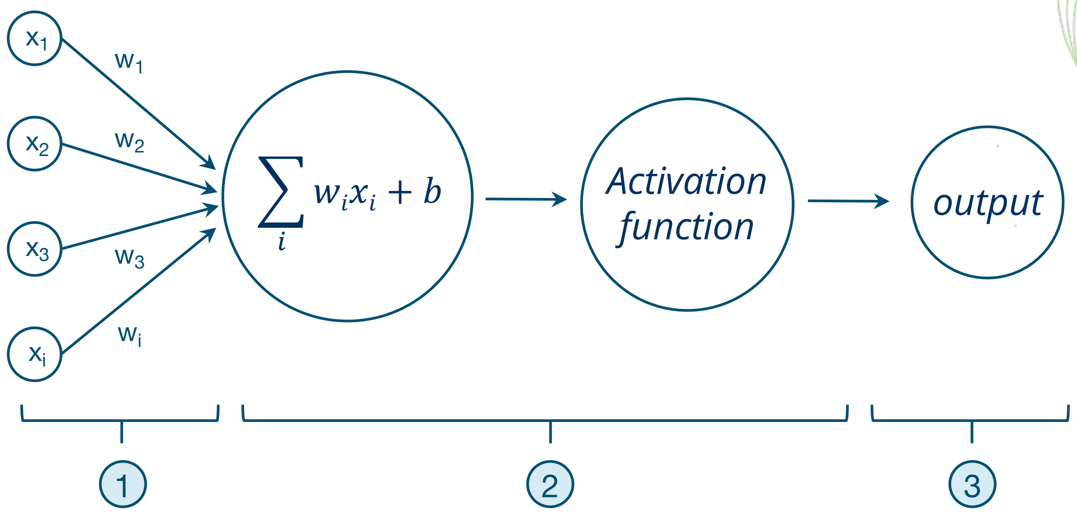
2024


ATARAXIS Homepage Technology Ataraxis Breast
Transforming Cancer Care with AI Precision Medicine

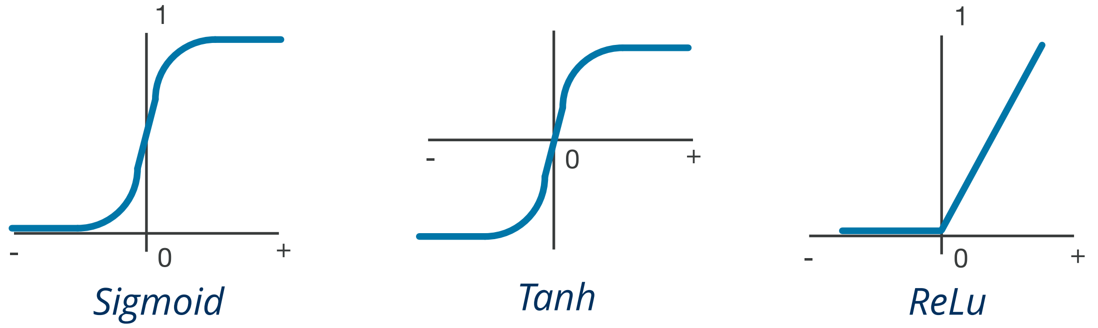
Basics of Neural Networks

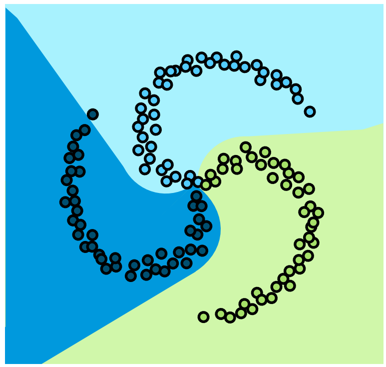
Theory





Activation functions:

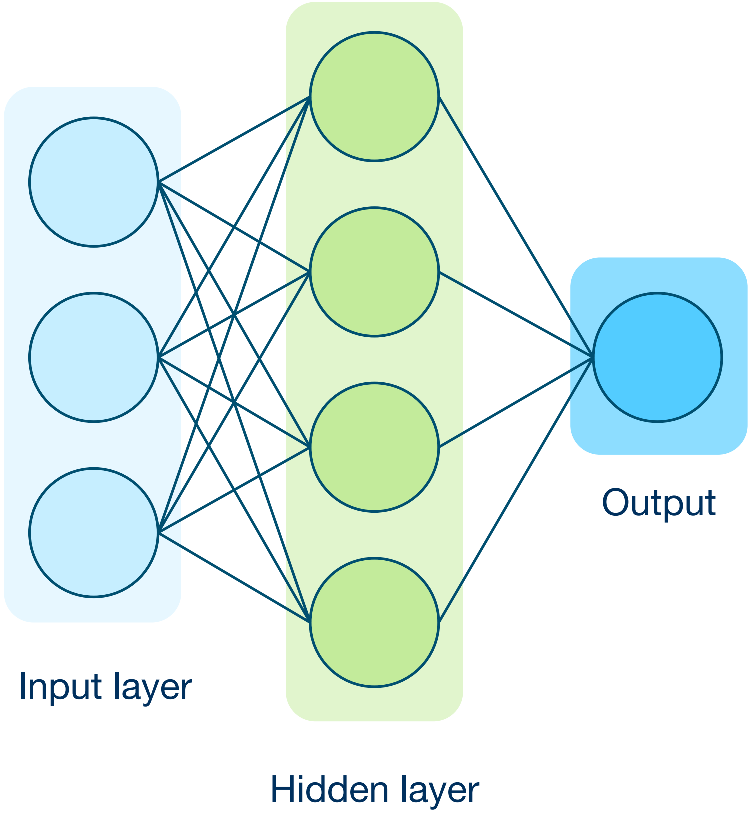


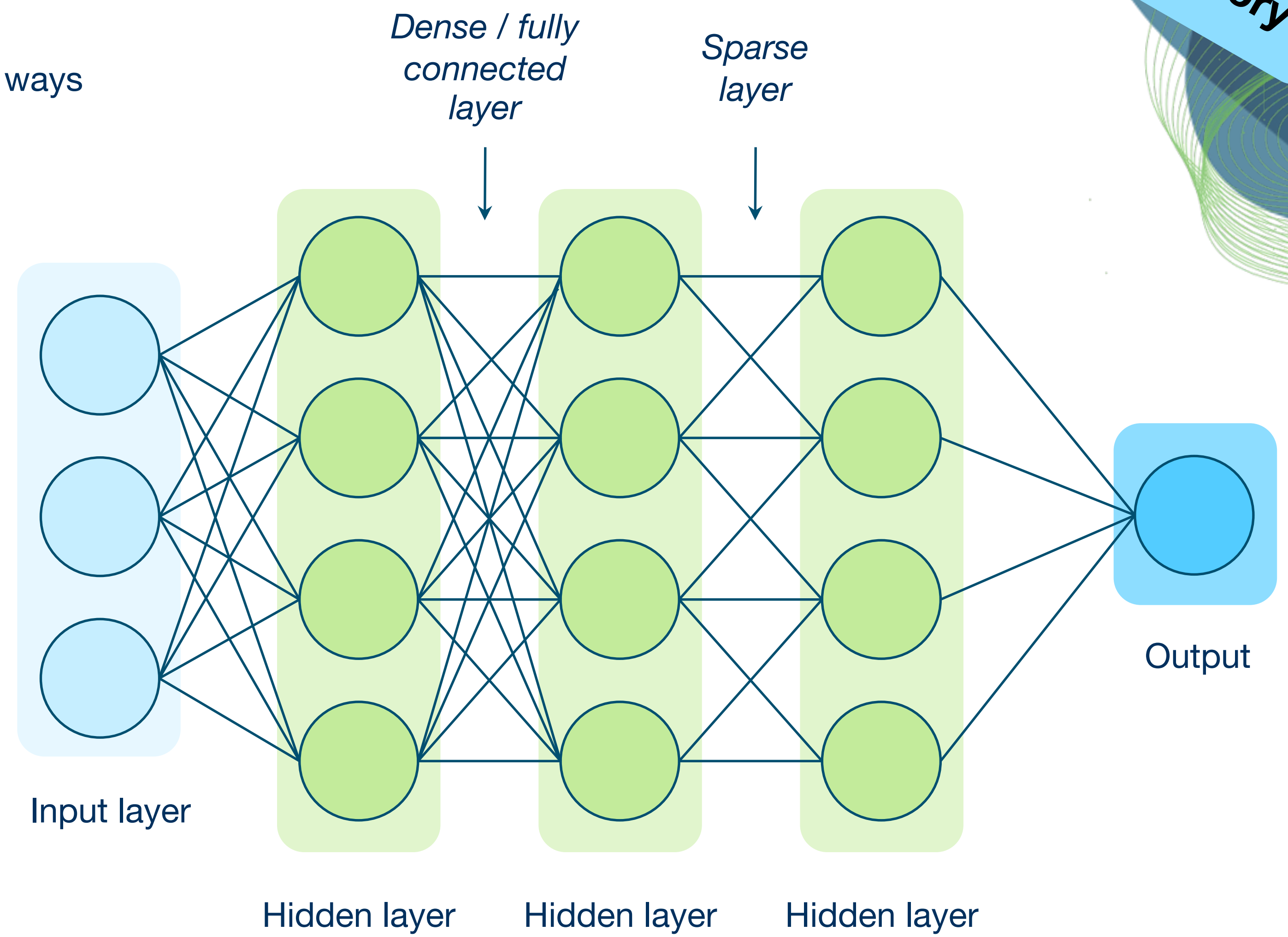


Activation functions introduce non-linearity to neural networks

Deep Neural Networks

- Artificial neurons can be stacked together in various ways





Pytorch

Theory

PyTorch

PyTorch is a Python framework for tensor computation and deep leaning.

- **tensor** = multi-dimensional array, can be scalar (0D), vector (1D), matrix (2D), ...
- Tensors are the **main data structure** used in deep learning. **Everything** in a model is **represented** as a tensor: input data, model parameters, model activations, gradients, ...
- In `scikit-learn` we hide the model in `fit()`. PyTorch makes it possible to **define** all **properties** of the model with the help of predefined classes/functions. This makes models more **flexible** and **customizable**.

Deep Neural Networks

Practice

1. Open the notebook **01_neural_networks.ipynb**
2. Load a built-in **breast cancer dataset** using `sklearn.datasets`

   (e.g. `load_breast_cancer()` for binary classification)
3. Split data into **train / validation / test** sets using `sklearn.model_selection`
4. Standardize features using `sklearn.preprocessing.StandardScaler`
5. Convert arrays to **PyTorch tensors** with `torch.tensor`
6. Create `TensorDataset` and `DataLoader` objects for mini-batch training

Deep Neural Networks – Data Loading

Practice

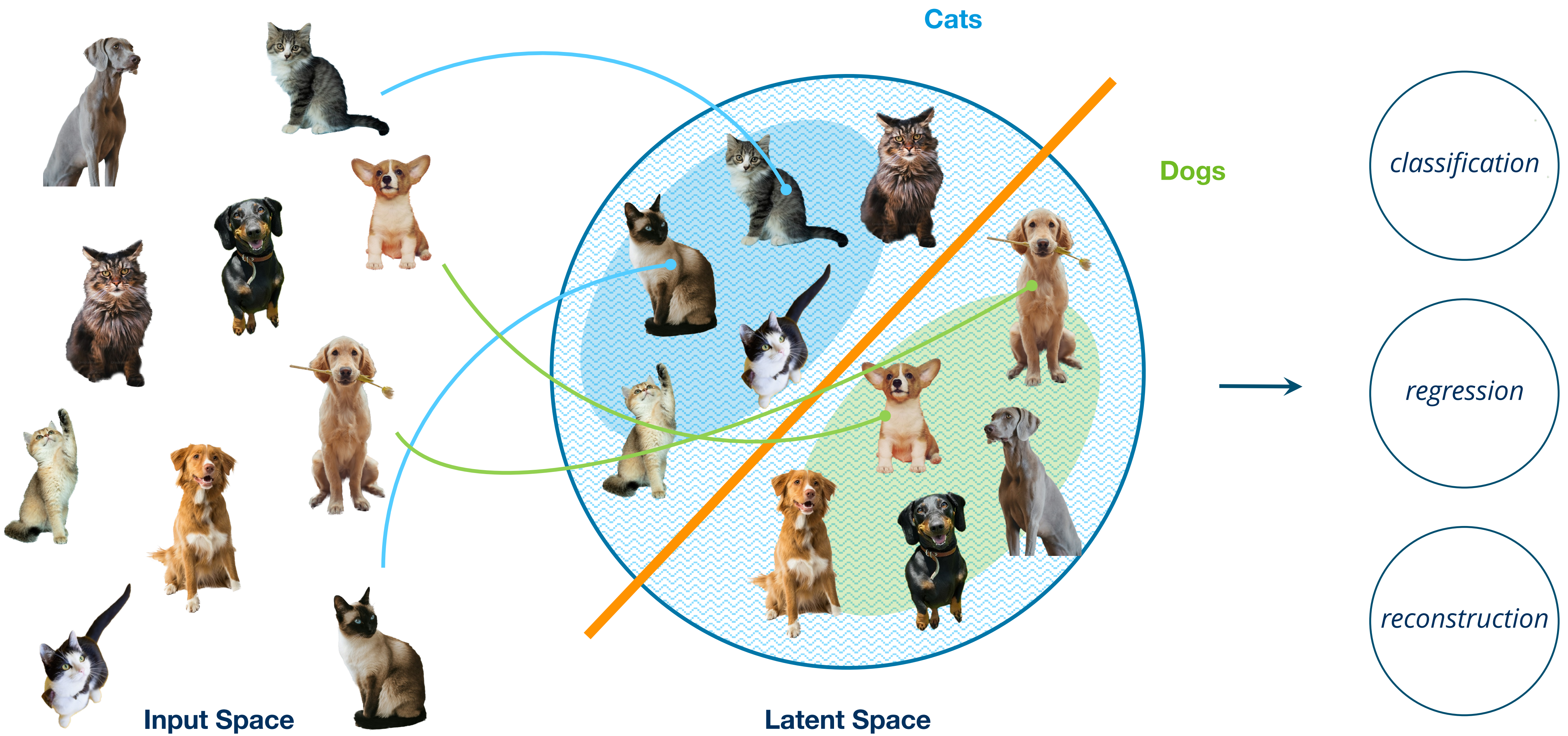
7. Implement a **PyTorch** model as a class inheriting from `nn.Module`

8. Define the **network architecture** in `__init__():`
   - input layer (number of features)
   - hidden layers (`nn.Linear`)
   - activation functions (`nn.ReLU, nn.Tanh`)
   - optional regularization (`nn.Dropout`)
   - output layer (1 node for binary classification)

9. Implement the **forward pass** in `forward(self, x)`

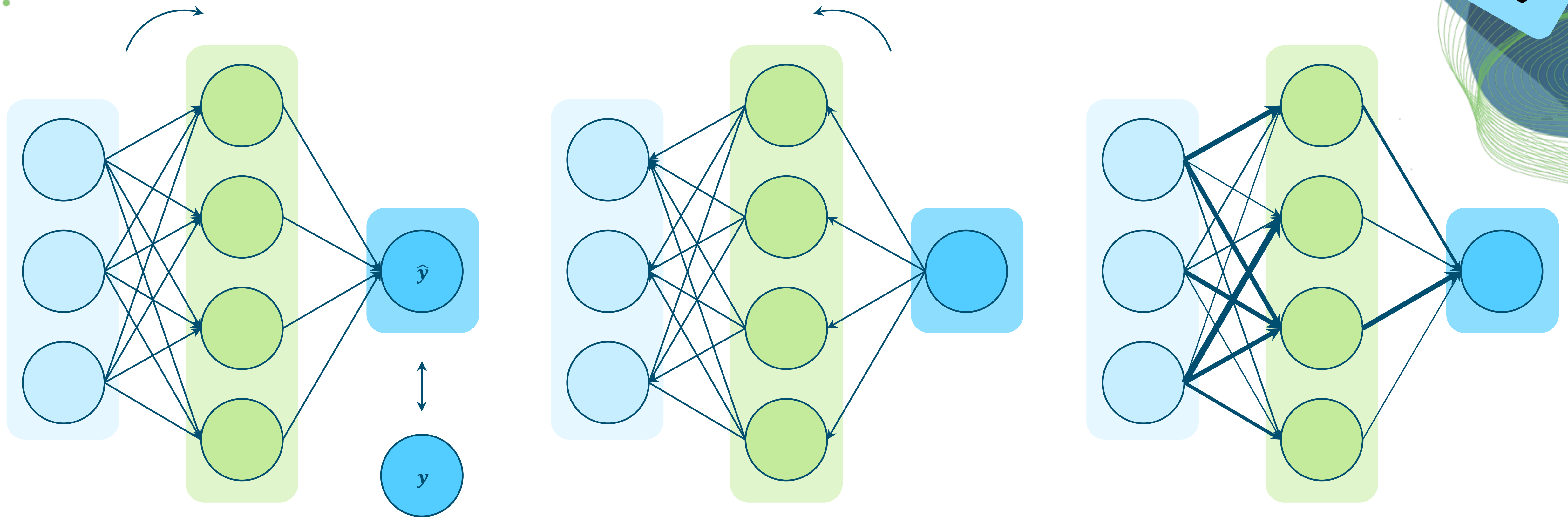
10. Instantiate the model with configurable **parameters**:
    - number of hidden layers
    - number of nodes per layer
    - activation function
    - dropout rate

The Latent Space



How do Neural Networks learn ?

Theory



1 ***Error***: *difference between prediction and output*

2 *Error is sent back through all neurons*

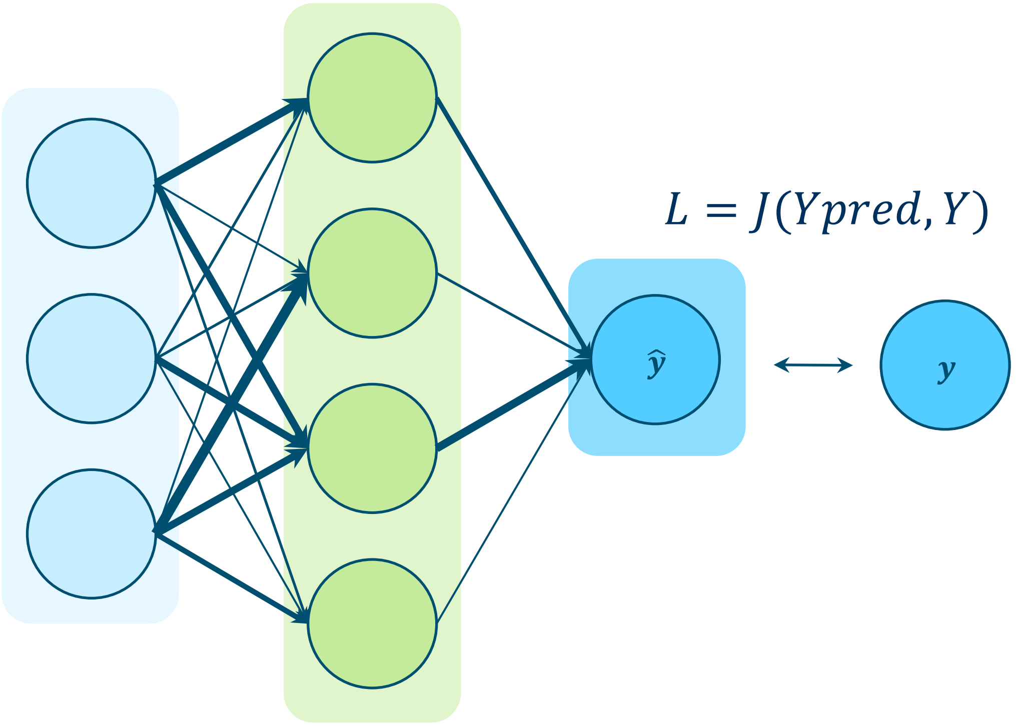
3 *Gradient of error is calculated for each weight*

4 *Weights are updated in regard to error*

5 *New error is sent back …*

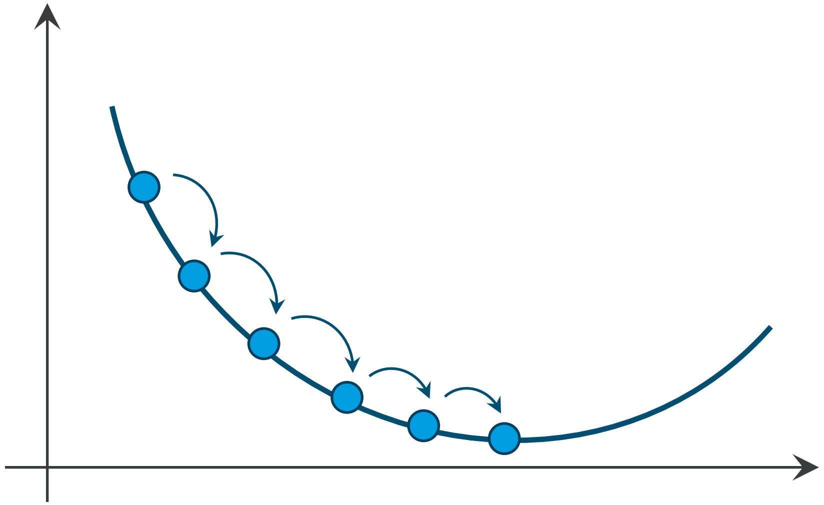
How do Neural Networks learn ?

Theory



Loss Function

- We want to minimize loss function and with this decrease the error between prediction and label.



Loss Function for Regression

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

Loss Binary Classification

$$L = -\frac{1}{N}\sum_{i=1}^{N}[yi\,\log(\hat{y}_i) + (1 - yi)\log(1 - \hat{y}_i)]$$

Loss Multi-Class Classification

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} log(\hat{y}_{ic})$$

How do Neural Networks learn ?

Practice

11. Define **loss function** and **optimizer**:
    - `nn.BCEWithLogitsLoss` for binary classification
    - `torch.optim.Adam` (or SGD)

12. Implement the **training loop**:
    - forward pass: `logits = model(x)`
    - compute loss
    - backpropagation: `loss.backward()`
    - update weights: `optimizer.step()`
    - reset gradients: `optimizer.zero_grad()`

13. Track **training and validation metrics** across epochs

14. Evaluate final performance on the **test set**

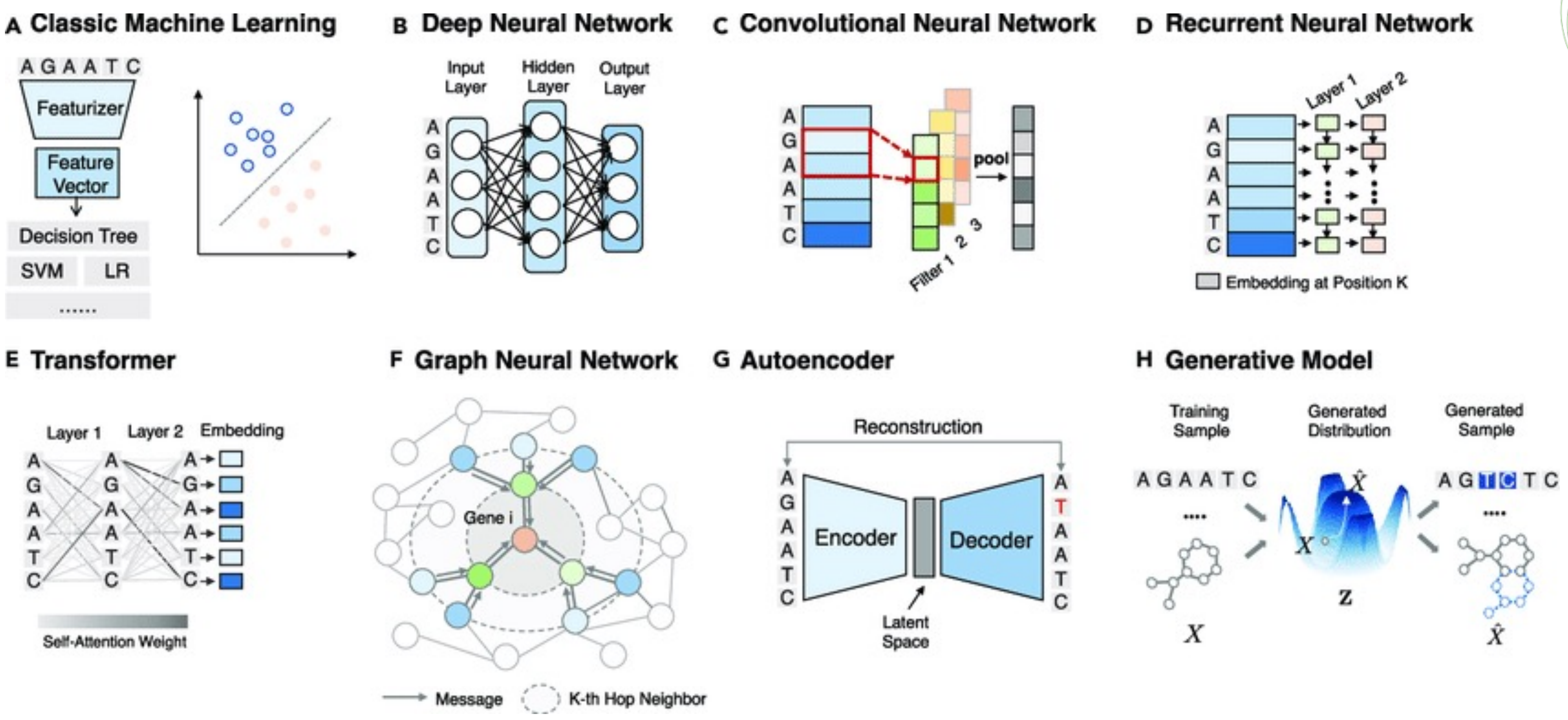
15. Compare different architectures and training settings

Any questions or remarks?

Exercise:

1. Change hidden layers / number of nodes / activation functions
2. Compare learning rates and number of epochs
3. Implement a 5-fold cross-validation to see how variable your results are

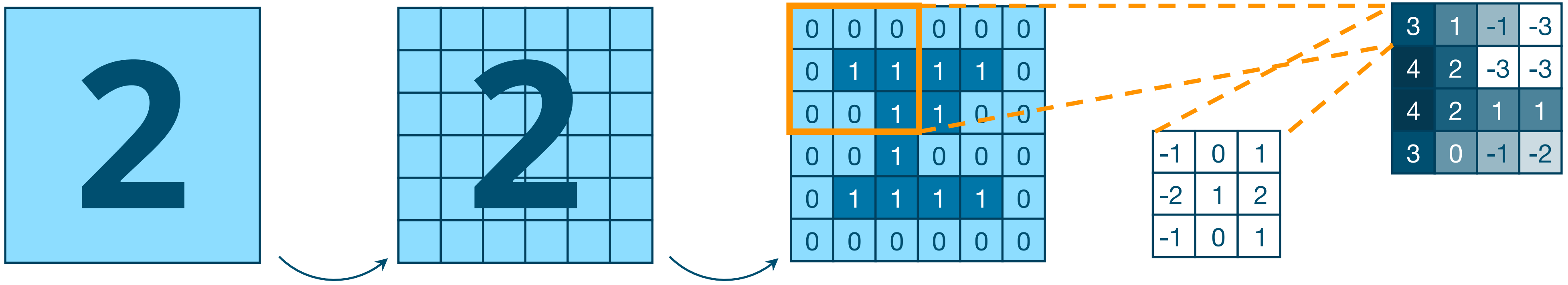
Todays Diversity



Theory

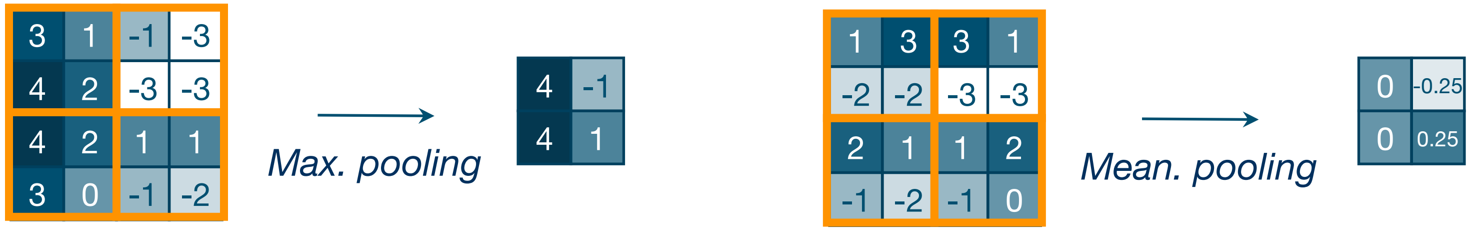
Example: Convolutional Neural Networks

Convolution:



- Filters detect structures within the image matrix

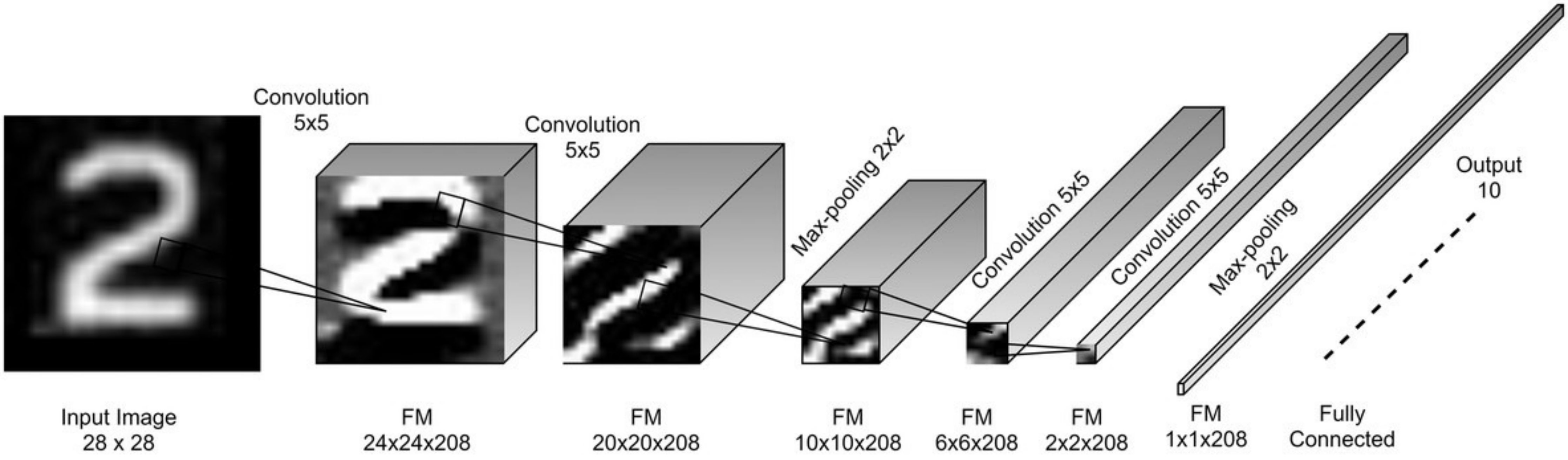
Pooling:

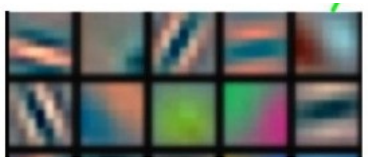


- Pooling reduces spatial size
- Makes features more robust to variation

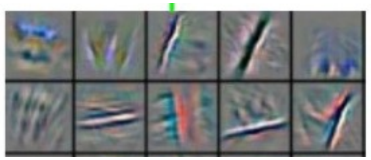
Example: Convolutional Neural Networks

Theory





Small-level features



Middle-level features

High-level features

output

classifier

GPUs

Theory

- Tensor operations are **compute heavy** (e.g. matrix multiplications, element-wise operations, …).
- Operations can be **parallelized** and and ran much **faster** on a GPU.
- Highly efficient for: image data, large datasets, large models, repeated experiments

When to use CPU:

Small data
Tabular data
No GPU available

When to use GPU:

images, videos
large models (e.g. CNNs, transformers, …)
LLM hosting

Convolutional Neural Networks

Practice

1. Open the notebook **02_convolutional_neural_networks.ipynb**

2. Load data and apply preprocessing

3. Implement the model as a custom PyTorch class that inherits from `torch.nn.Module`
   - `__init__()`, load a **pretrained CNN backbone** (e.g. **ResNet18**)
   - pretrained **ImageNet** weights so the model starts from useful image features
   - replace the final fully connected **classification layer** (`model.fc`) with a new layer for our task
   - Optionally: **freeze the backbone** (`requires_grad=False`) to train only the new classification head first or train it (`requires_grad=True`)
   - Implement the `forward(self, x)` method to define how input images pass through the network and produce class scores

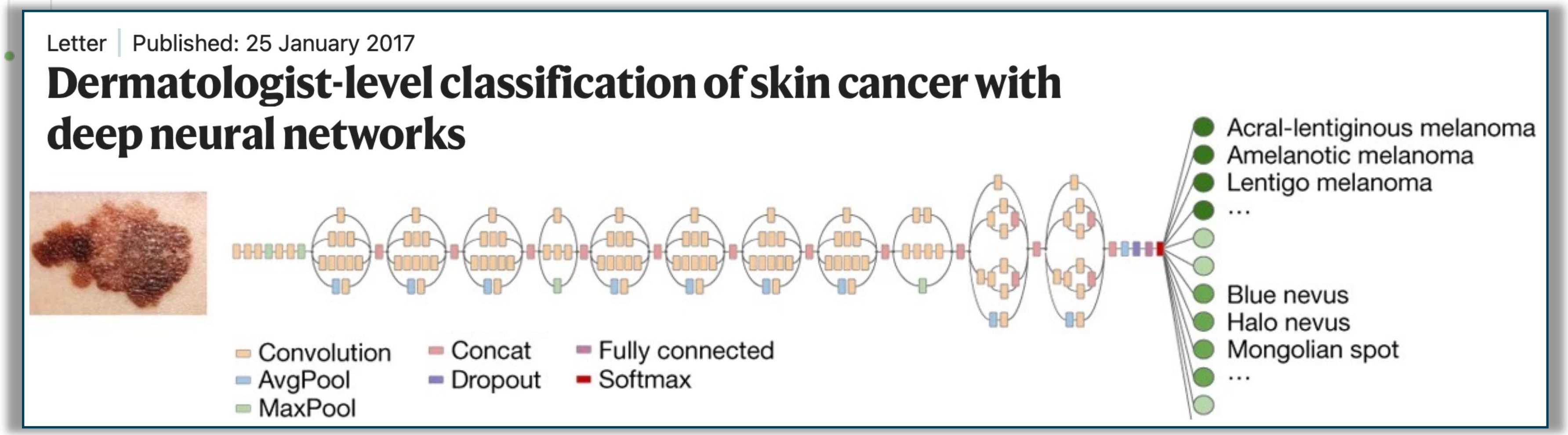
4. Train and validate the model

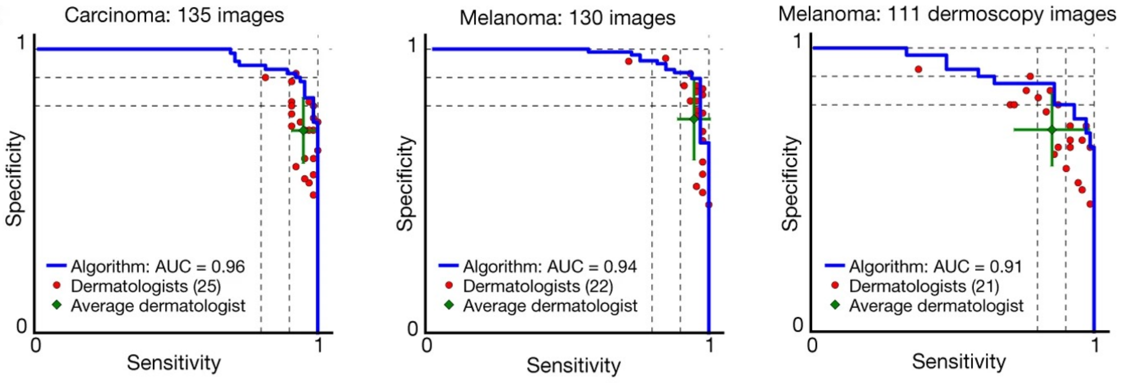
Any questions or remarks?

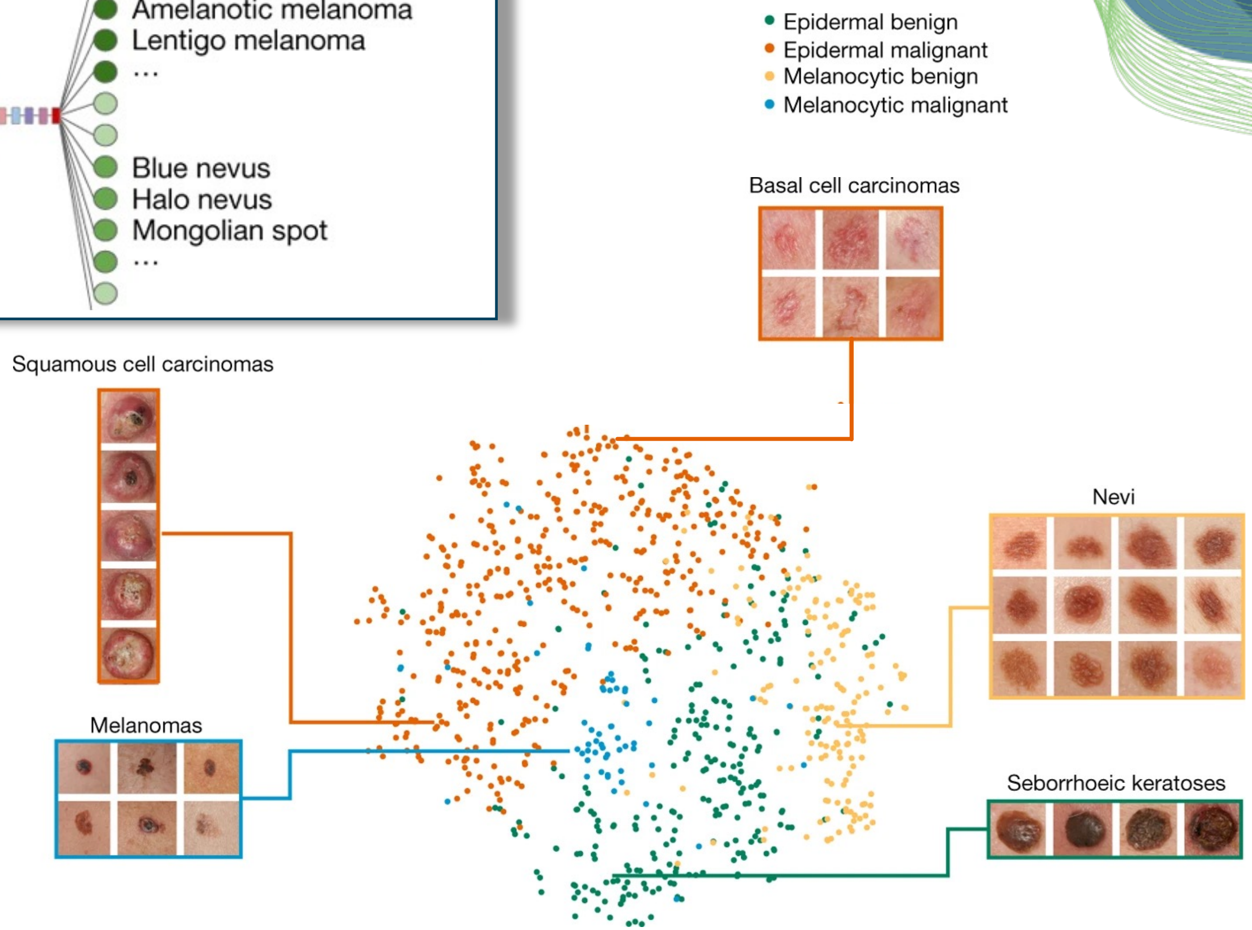
Exercise:

1. What is the difference between frozen and unfrozen backbones ? What happens in practice ?
2. Try out different ResNet backbones
3. Adapt code for breed classification

Example - Melanomas



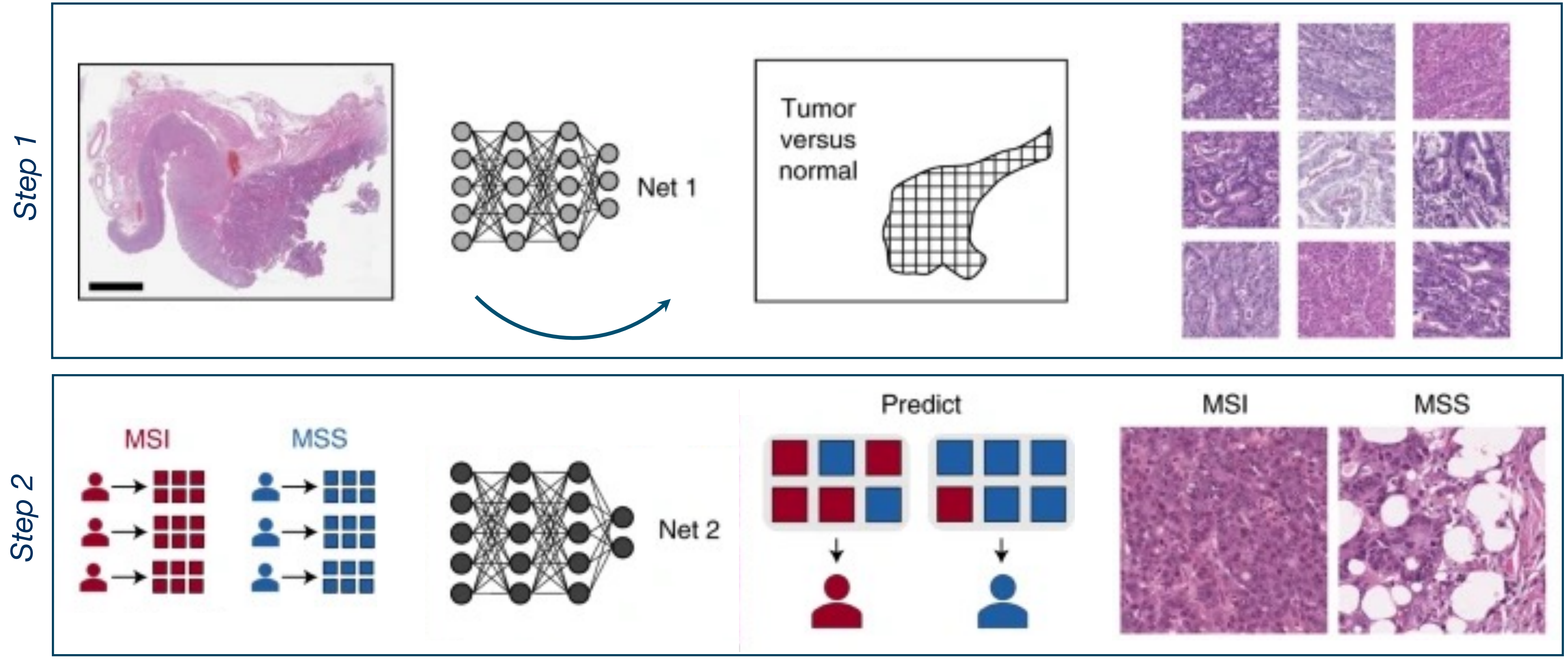


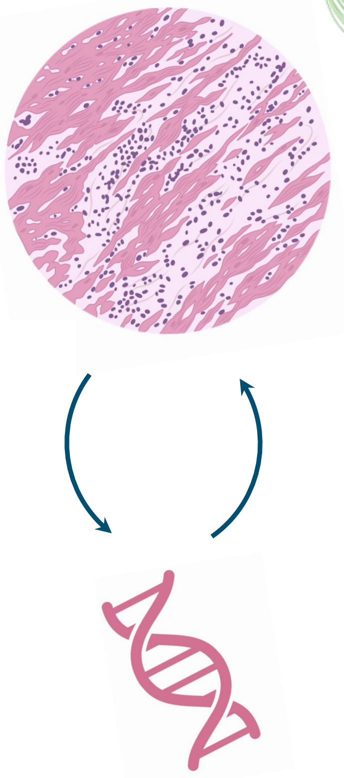


Example - Pathology

Brief Communication | Published: 03 June 2019

Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer

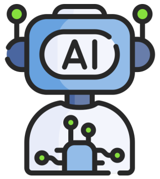




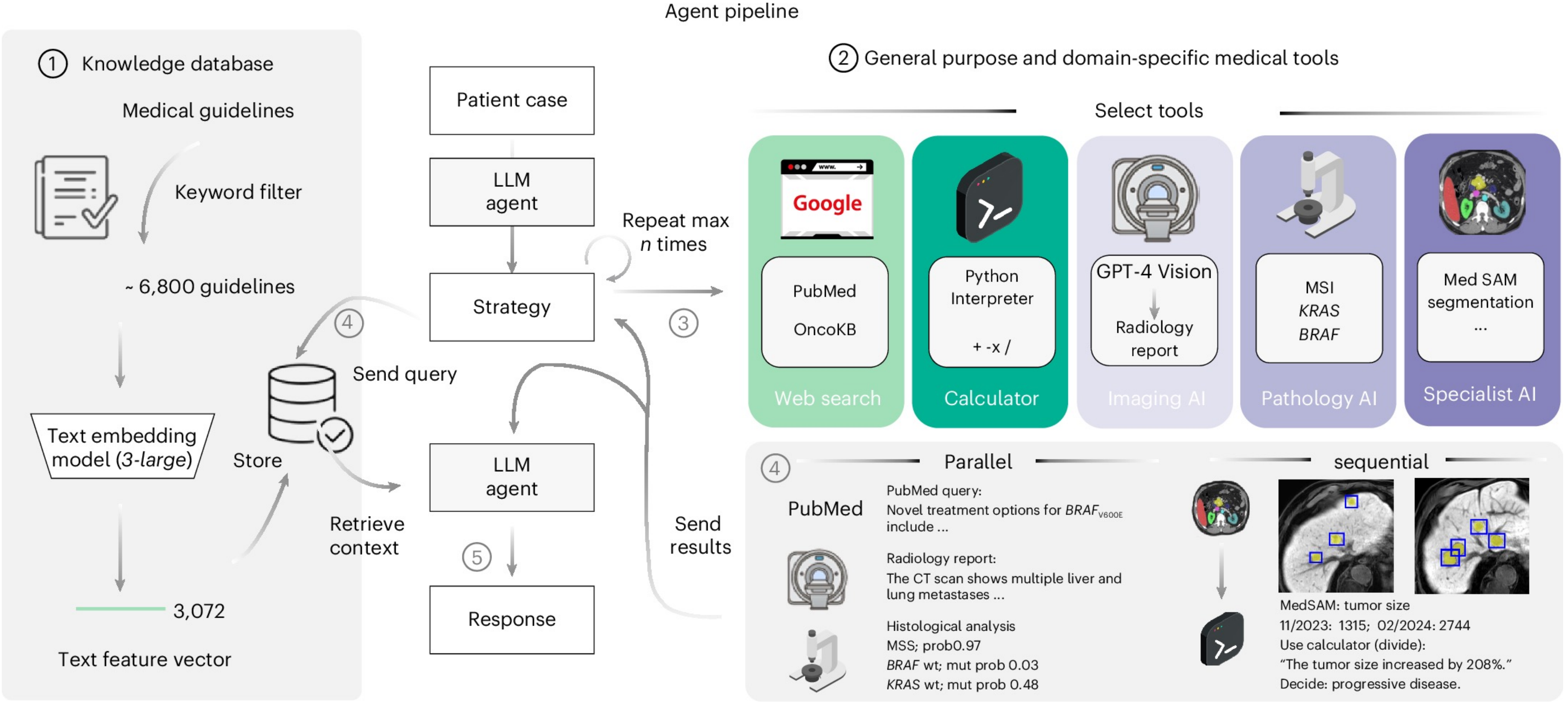
Outlooks – AI Agents

Article | Open access | Published: 06 June 2025

Development and validation of an autonomous artificial intelligence agent for clinical decision-making in oncology



AI agent



Challenges and Limitations

Data Challenges

- Bias in training data
- Imbalance of training data
- Data availability / open-access
- Artefacts

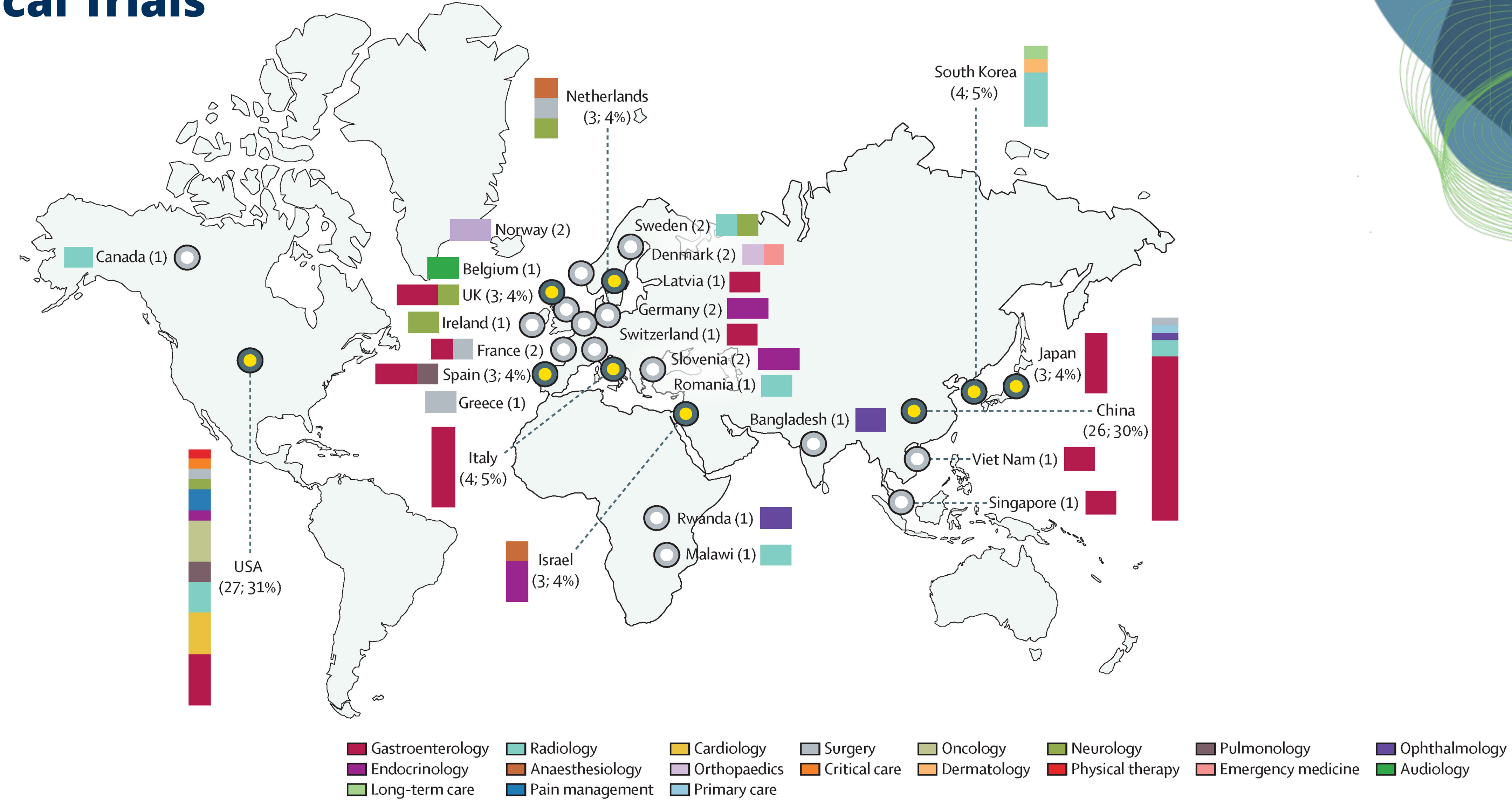
Model Challenges

- Digitization
- Explainability
- Decisions aligned with newest clinical guidelines
- Compute
- Size

Clinical Challenges

- Safe use in best interest of patients
- Performance degradation over time / domain shift
- Compliance with regulatory standards
- Easy and accessible use

Clinical Trials



Any questions or remarks?

Exercise:

Let’s classify some medical images !