Week 5

model: overfits on training data

world: new data

model:

Review

- Simplest network – perceptron
- Input -> Hidden layer -> Output layers
- Each neuron performs a weighted sum + bias -> activation
- The only type we have talked about is Feedforward (MLP)

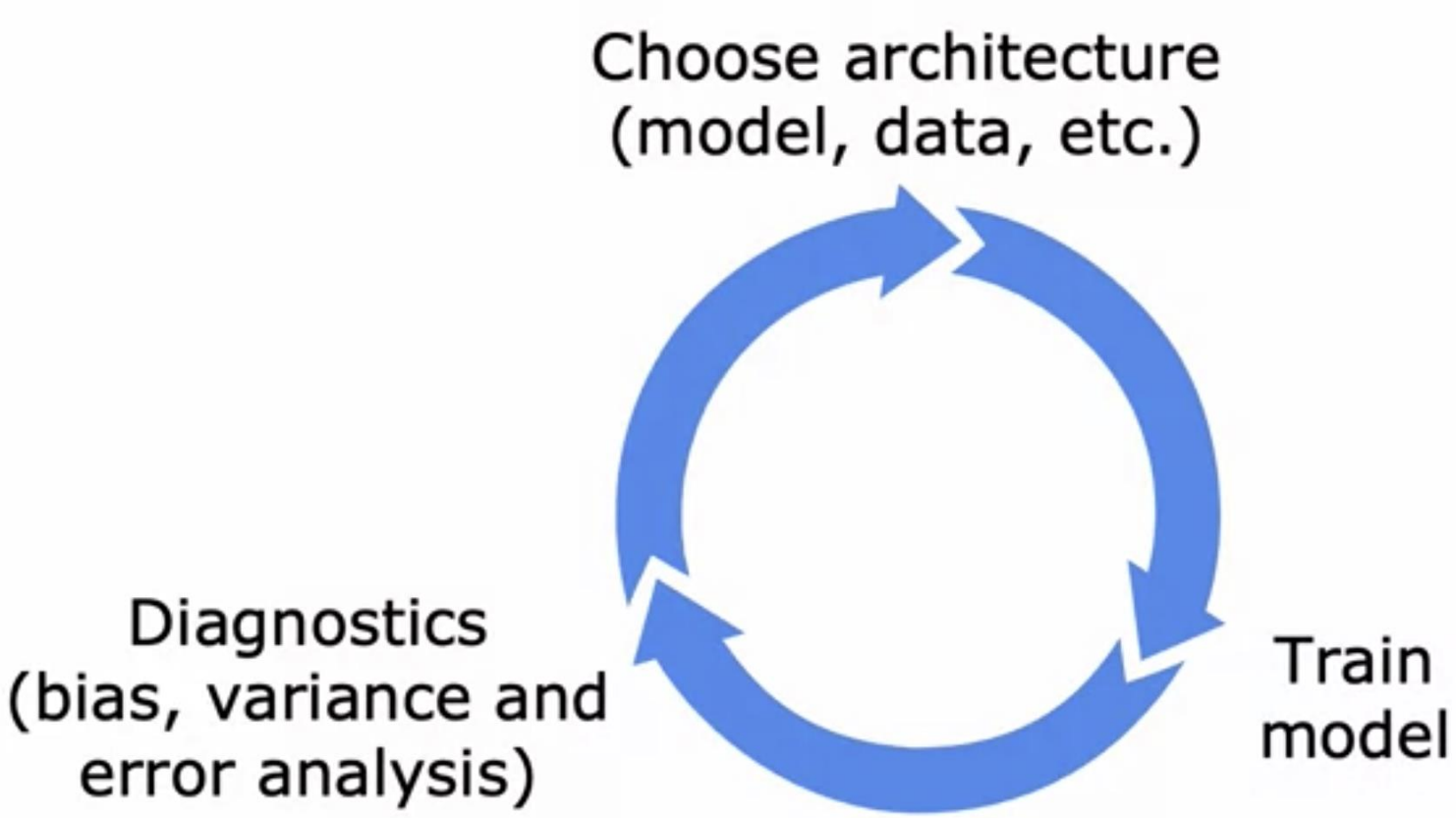
What to do if model overfits?

What to do if model overfits?

- Regularization
- Early stopping
- Dropout

Iterative loop of ML development

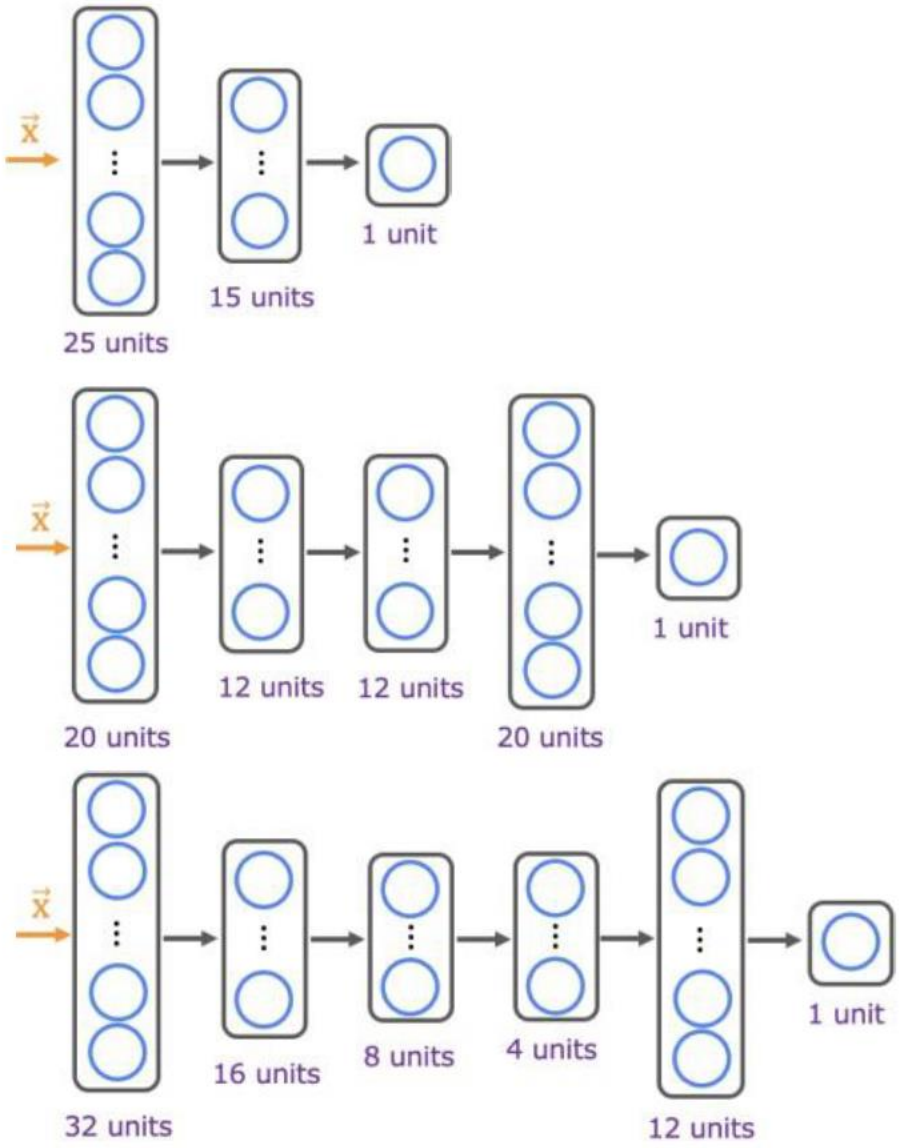
Data preprocessing
Feedfoward (MLP)



Trained MNIST classifier

Choosing a Neural Network Architecture

- Pick best neural net model using validation set error.
- Estimate generalization error using the test set error.



Debugging a Learning Algorithm

- When a model performs poorly, start by diagnosing the issue rather than guessing.

Common fixes:

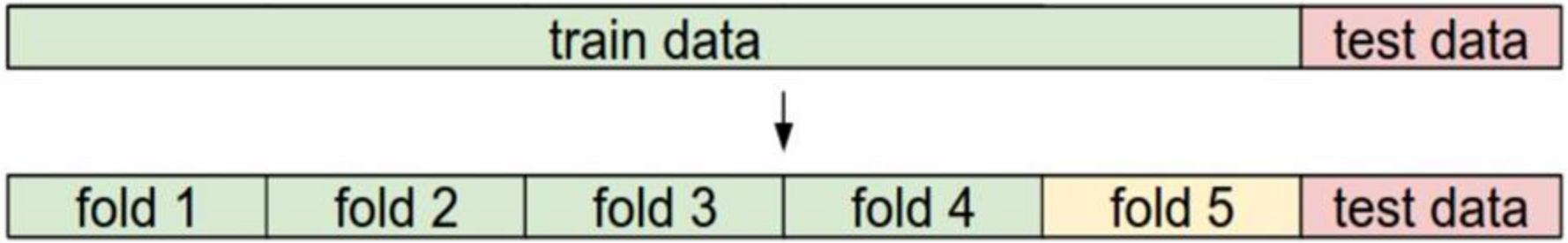
- Add more training data (reduces variance)
- Reduce features (simplifies model)
- Add new features (improves representation)
- Add polynomial/interactions (reduces bias)
- Tune regularization

Model Evaluation – Train/Test Split

- Split your data to evaluate generalization (70/15/15)
- Avoid data leakage

Model Evaluation - Cross Validation

- Avoid selecting models based directly on the test set.
- Use a validation set or k-fold cross validation.



Common data splits. A training and test set is given. The training set is split into folds (for example 5 folds here). The folds 1-4 become the training set. One fold (e.g. fold 5 here in yellow) is denoted as the Validation fold and is used to tune the hyperparameters. Cross-validation goes a step further and iterates over the choice of which fold is the validation fold, separately from 1-5. This would be referred to as 5-fold cross-validation. In the very end once the model is trained and all the best hyperparameters were determined, the model is evaluated a single time on the test data (red).

Evaluation Metrics Overview

Model evaluation depends on task type (classification, regression, ranking, etc.)

Regression metrics:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)

Classification metrics:

- Accuracy, Precision, Recall, F1-score
- ROC-AUC, PR-AUC
- Recommendation systems:
- Hit rate, MAP@k, NDCG

Regression Metrics

- MAE: more robust to outliers
- MSE: penalizes large errors more (useful when big deviations are costly)
- RMSE: penalized large errors mor
- R^2: measures how well the mod explains the variability of the targ variable

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

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

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

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ – *predicted value of* y

$\bar{y}$ – *mean value of* y

Classification Metrics: Confusion Matrix

- Helps identify type of errors the model makes.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

		Predicted Condition			
		Predicted Positive (PP)	**Predicted Negative (PN)**		
Actual Condition	**Actual Positive (P)**	**True Positive (TP)** "Hit" "Correctly rejected null hypothesis"	**False Negative (FN)** "Type II error", "Miss" "Incorrectly kept null hypothesis"	**True Positive Rate (TPR)** "**Recall**", "**Sensitivity**", "**Power**", "Completeness" $= \frac{TP}{P} = 1 - FNR$ $= P(\hat{Y} = 1 \mid Y = 1)$	**False Negative Rate (FNR)** "Miss rate" $= \frac{FN}{P} = 1 - TPR$ $= P(\hat{Y} = 0 \mid Y = 1)$
	Actual Negative (N)	**False Positive (FP)** "Type I error", "False alarm" "Incorrectly rejected null hypothesis"	**True Negative (TN)** "Correct rejection" "Correctly kept null hypothesis"	**False Positive Rate (FPR)** "Probability of false alarm" $= \frac{FP}{N} = 1 - TNR$ $= P(\hat{Y} = 1 \mid Y = 0)$	**True Negative Rate (TNR)** "**Specificity**" $= \frac{TN}{N} = 1 - FPR$ $= P(\hat{Y} = 0 \mid Y = 0)$
		Positive Predictive Value (PPV) "**Precision**", "Exactness" $= \frac{TP}{PP} = 1 - FDR$ $= P(Y = 1 \mid \hat{Y} = 1)$	**False Omission Rate (FOR)** $= \frac{FN}{PN} = 1 - NPV$ $= P(Y = 1 \mid \hat{Y} = 0)$		
		False Discovery Rate (FDR) $= \frac{FP}{PP} = 1 - PPV$ $= P(Y = 0 \mid \hat{Y} = 1)$	**Negative Predictive Value (NPF)** $= \frac{TN}{PN} = 1 - FOR$ $= P(Y = 0 \mid \hat{Y} = 0)$		
		Total Population $= P + N = (TP + FN) + (TN + FP)$ $= PP + PN = (TP + FP) + (TN + FN)$	**Prevalence** "Proportion affected" $= \frac{P}{P + N}$	**Accuracy** $= \frac{TP + TN}{P + N}$	**F_1 Score** $= harmonic\ mean(precision, recall)$ $= \frac{2PPV * TPR}{PPV + TPR}$
			Matthews correlation coefficient (MCC) $= \sqrt{TPR * TNR * PPV * NPV} - \sqrt{FNR * FPR * FOR * FDR}$ (Considered most balanced, single metric to evaluate confusion matrix)	**Balanced Accuracy** $= \frac{TPR + TNR}{2}$	*(Avg metrics of current and flipped pos/neg labels; allows looking at P/R at ALL classes)* **Macro-Avg-Precision** $= (PPV + NPF)/2$ **Macro-Avg-Recall** $= (TPR + TNR)/2$ **Macro-Avg-F1** $= \left(\frac{2PPV*TPR}{PPV+TPR} + \frac{2NPF*TNR}{NPF+TNR}\right)/2$

Classification Metrics: Confusion Matrix

Actual	Predicted
Cat	Cat
Cat	Cat
Cat	Cat
Cat	Dog
Dog	Dog
Dog	Dog
Dog	Cat

Classification Metrics: Confusion Matrix

Actual	Predicted
Cat	Cat
Cat	Cat
Cat	Cat
Cat	Dog
Dog	Dog
Dog	Dog
Dog	Cat

	Predicted: Cat	**Predicted: Dog**
Actual: Cat	TP = 3	FN = 1
Actual: Dog	FP = 1	TN = 2

Classification Metrics: Confusion Matrix

Actual	Predicted
Cat	Cat
Cat	Cat
Cat	Cat
Cat	Dog
Dog	Dog
Dog	Dog
Dog	Cat

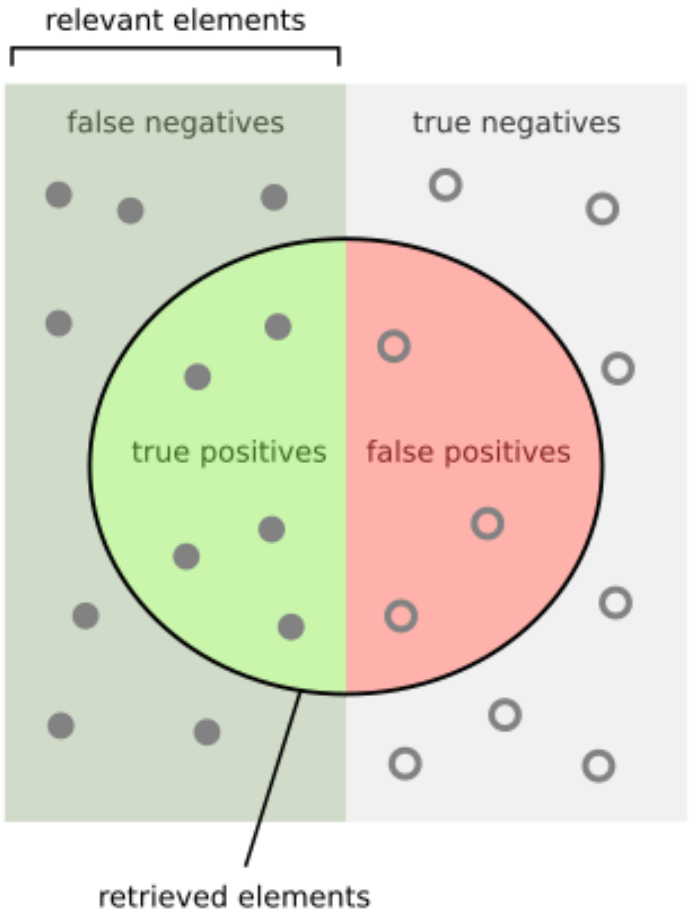
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3 + 2}{3 + 2 + 1 + 1} = \frac{5}{7} \approx 0.714$$

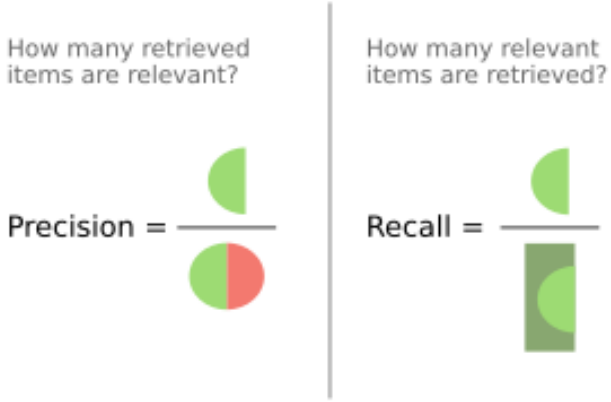
	Predicted: Cat	**Predicted: Dog**
Actual: Cat	TP = 3	FN = 1
Actual: Dog	FP = 1	TN = 2

Classification Metrics

- Accuracy: fraction of correct predictions (**useful only for balanced classes)**
- Precision: fraction of predicted positives that are correct
  - $\underline{\text{Positive Predictive Value = Precision}} = PPV = \frac{TP}{(TP + FP)}$
- Recall: fraction of actual positives correctly predicted
  - $\underline{\text{True Positive Rate = Recall}} = TPR = \frac{TP}{(TP + FN)}$
- F1-score: harmonic mean of precision and recall

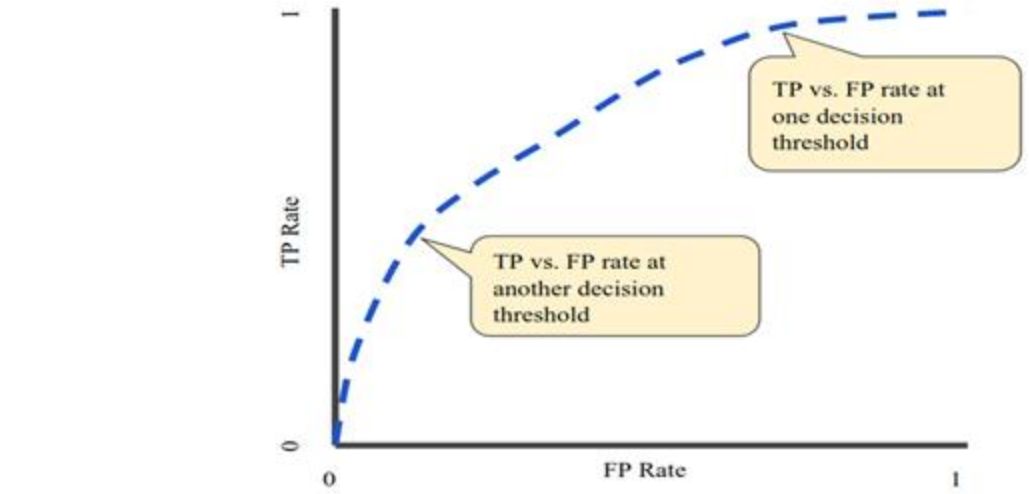
$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

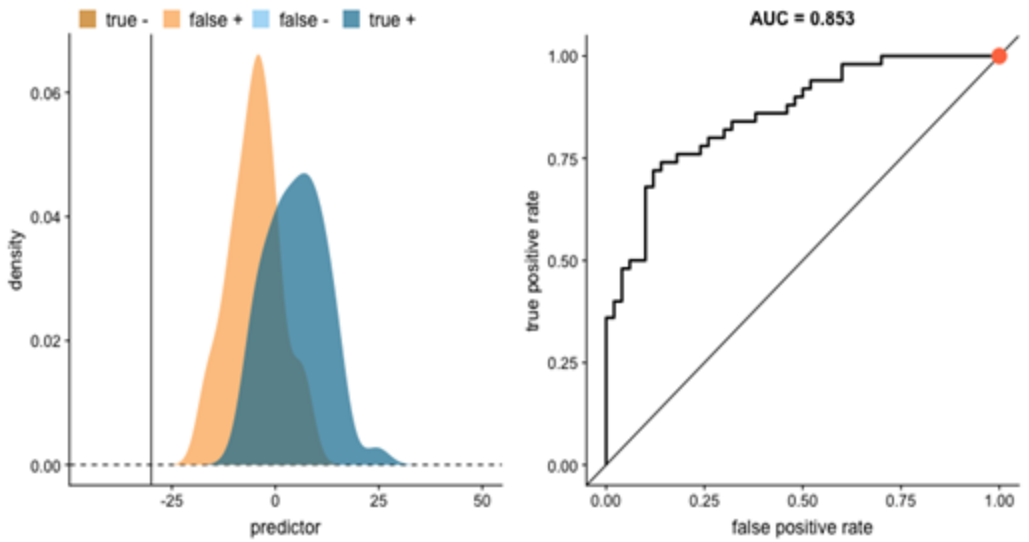




Classification metrics: ROC Curve

- ROC (Receiver Operating Characteristic) Curve: plots TPR vs. FPR for different thresholds
  - Each point in the ROC curve thus represents a distinct confusion matrix for some threshold value
- AUC (area under the curve) of the ROC curve: a summary of the performance of a method
  - It measures ability to rank positives above negatives





Introduction to Convolutional Neural Networks

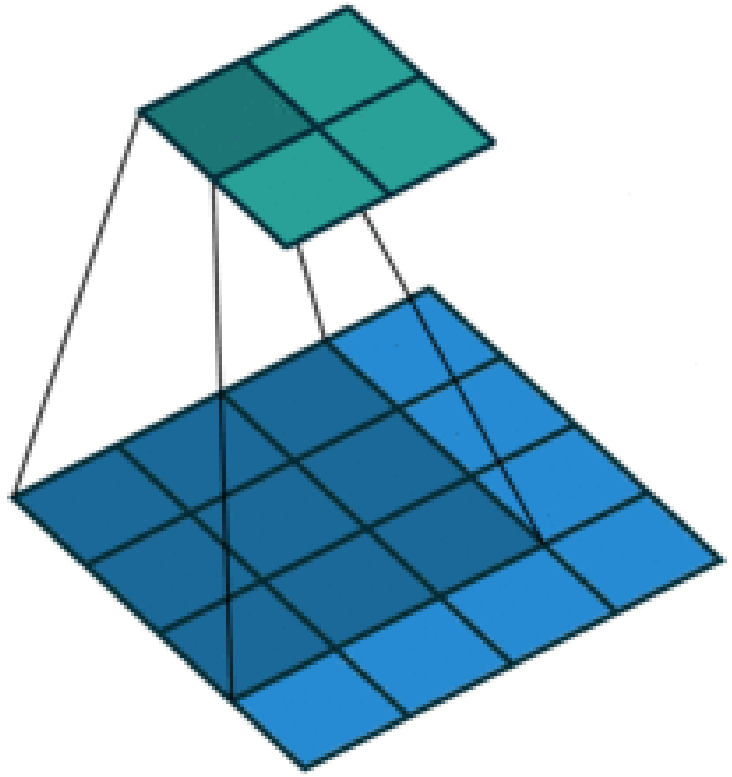
- Neural networks learn patterns from data for classification, prediction, and generation.
  - CNNs excel at spatial data like images.

- CNNs extract hierarchical spatial features using filters and convolutions.

Introduction to Convolutional Neural Networks

Key Components:

- Convolutional Layer: Extracts local spatial patterns.
- ReLU: Adds non-linearity.
- Pooling: Reduces dimensionality while keeping important features
- Fully Connected: Combines features for final classification.



Applications:

- Medical imaging (detecting tumors in CT scans).
- Self-driving cars (lane and object detection).
- Face recognition and biometric systems.

CNN in PyTorch

Example Implementation:

```python
import torch.nn as nn
cnn = nn.Sequential(
  nn.Conv2d(3, 32, 3, padding=1),
  nn.ReLU(),
  nn.MaxPool2d(2),
  nn.Conv2d(32, 64, 3, padding=1),
  nn.ReLU(),
  nn.Flatten(),
  nn.Linear(64 * 16 * 16, 128),
  nn.ReLU(),
  nn.Linear(128, 10)
)
```

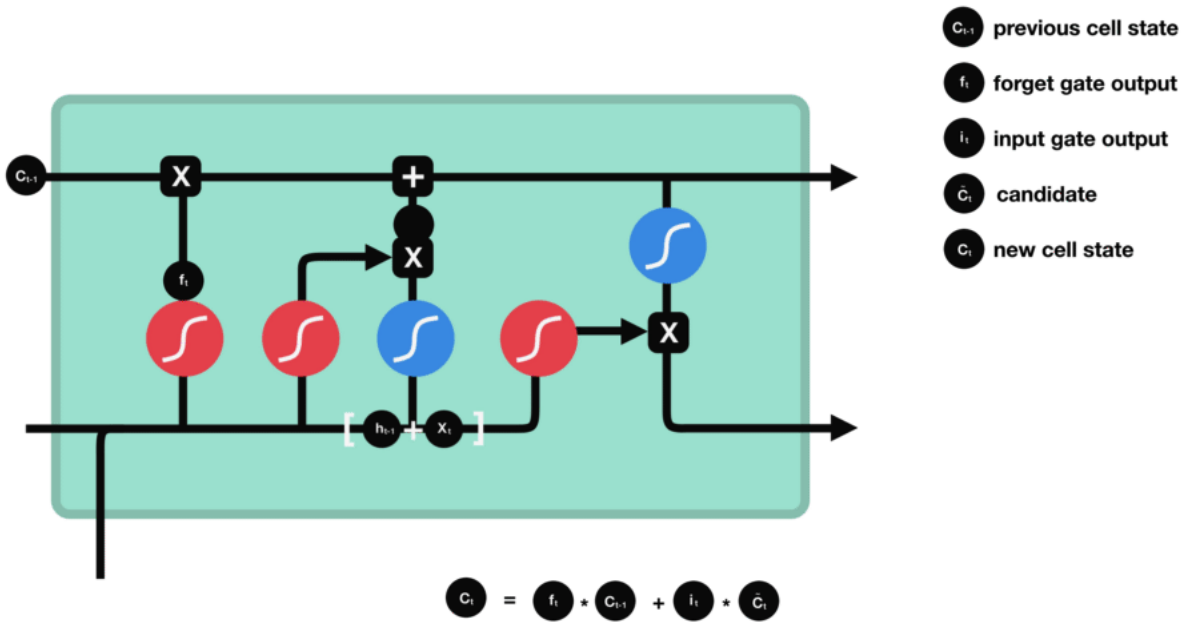
Introduction to Reccurent Neural Networks

Concept:

RNNs process sequences one step at a time, maintaining memory of temporal connections.

Variants:

- Simple RNN: Basic recurrence for short sequences.
- LSTM: Long Short-Term Memory — remembers long-term dependencies.
- GRU: Gated Recurrent Unit — faster, simplified LSTM.



Applications:

- Text generation and translation.
- Speech recognition.
- Time-series forecasting (finance, weather, healthcare).

RNN in PyTorch

- Example Implementation:

```
import torch.nn as nn
rnn = nn.LSTM(input_size=50, hidden_size=100, num_layers=2, batch_first=True)
output, hidden = rnn(input_seq)
```