

Spring 2025

Introduction to Artificial Intelligence

Homework 3 – Supervised Learning

Due Date: 2025/04/21 17:00

Introduction

Convolutional Neural Networks (CNN)

This computer assignment involves 1) designing a CNN for object recognition, 2) training a CNN given a dataset, and 3) testing the trained CNN on novel images.

CNN is a type of artificial neural network composed of multiple convolutional, pooling, and fully-connected layers. The goal of a CNN is to learn higher-order features in the data via convolutions. They are well suited to object recognition with images and consistently top image classification competitions. They can identify faces, individuals, street signs, platypuses, and many other aspects of visual data. CNNs overlap with text analysis via optical character recognition, but they are also useful when analyzing words as discrete textual units. They're also good at analyzing sound.

Let's now take a look at the key components of CNNs.

- **Convolutional Layer:** Extracts features from the input image using small filters that detect edges, textures, and shapes. It is the core building block of a CNN.
- **ReLU:** Commonly applied after a convolutional layer. It helps CNNs learn complex patterns and improves training stability by reducing the vanishing gradient problem.
- **Pooling Layer:** Reduces the size of feature maps while keeping important information, making the network faster and less prone to overfitting.
- **Fully Connected Layer:** Flattens the extracted features and connects them to a final classification layer, is typically found near the end of a CNN.

Fig. 1 depicts the classification process of CNN.

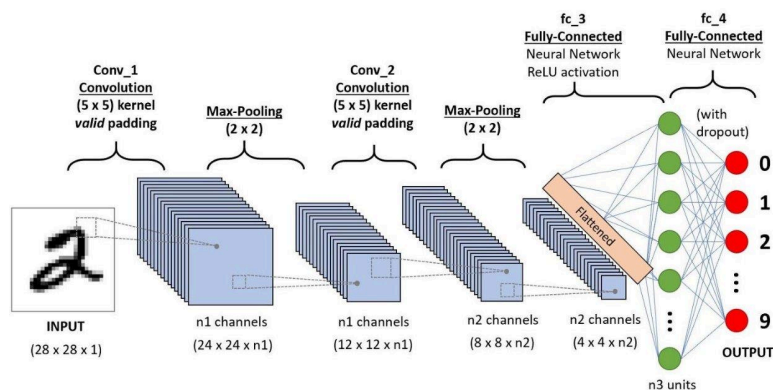


Fig 1. Schematic diagram of the convolution neural network classification process.

Decision Tree

This computer assignment involves 1) using a pretrained CNN to extract the feature of images and 2) classifying images using the Decision Tree model. (Fig. 2)

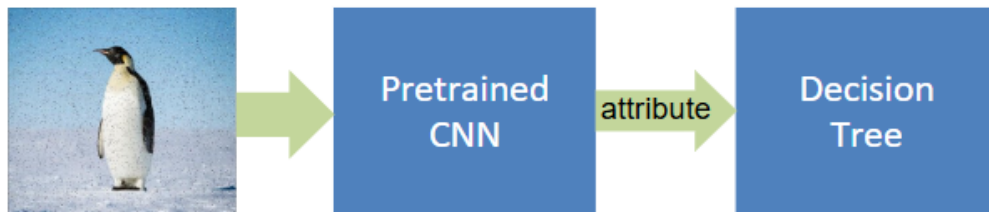


Fig 2. Schematic diagram of image classification using Pretrained CNN and Decision Tree.

A decision tree (generally defined) is a tree whose internal nodes are tests (on input features) and whose leaf nodes are categories (of features) (Fig. 3). A decision tree is created by recursively partitioning the data into smaller and smaller subsets. At each partition, the data is split based on a specific feature, and the split is made in a way that optimizes a chosen metric, such as information gain (which reduces uncertainty or entropy) or Gini impurity, depending on the implementation.

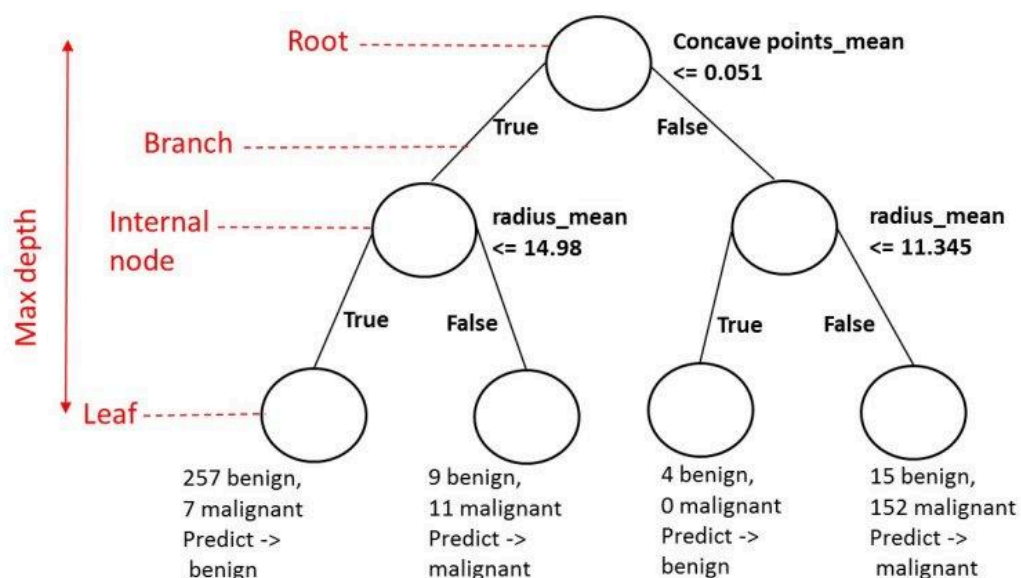


Fig. 3 A Decision Tree

Implementation and Report (60%)

Please implement the following code **using PyTorch** as specified.

For each part, take clear screenshots of your code and include them in your report.

Also, provide detailed explanations for each segment of the code.

Part 1: CNN

First, define functions in `utils.py` to load images.

- Data Loading (0%)

- **load_train_dataset()**

- Load the training dataset from the given path 'data/train/'.
The function reads images and assigns labels based on the folder name of each animal.

- **You should assign labels defined as follows:**

```
{0: "elephant", 1: "jaguar", 2: "lion", 3: "parrot",  
4: "penguin"}
```

- Returns: (images, labels)
images: The list of all images.
labels: The corresponding labels for each image.

- **load_test_dataset()**

- Load testing dataset from the given path 'data/test/'
 - Returns: images

Next, design your CNN and define function train() and validate() in **CNN.py**

- Design Model Architecture (10%)

- **__init__()**

- Design your CNN, it can only be **less than 3 convolution layers**

- **forward()**

- Define the forward pass of the CNN, ensuring it corresponds to the architecture in **__init__()**.

- Define function train(), validate() and test() (10%)

- **train()**

- Train the CNN model on the training dataset and compute the average loss.
 - We suggest use tqdm to know the progress
 - Returns: avg_loss (float): The average loss across all batches during training.

- **validate()**

- Validate the CNN model on the validation dataset and compute the average loss and accuracy.
 - We suggest use tqdm to know the progress
 - Returns:
 - avg_loss (float): The average loss across all batches during validation.
 - accuracy (float): The overall accuracy on the validation dataset.

- **test()**

- Test the model on testing dataset and write the result to 'CNN.csv'

- Save results to 'CNN.csv' in the following format:
 - Column 'id': image name.
 - Column 'prediction': Predicted class for each test image.

Then, print the training log to help you monitor the training process in `main.py`

- Printing Training Logs (0%)

```
for epoch in range(EPOCHS): #epoch
    train_loss = train(model, train_loader, criterion, optimizer, device)
    val_loss, val_acc = validate(model, val_loader, criterion, device)

    train_losses.append(train_loss)
    val_losses.append(val_loss)

    # (TODO) Print the training log to help you monitor the training process
    #         You can save the model for future usage
    raise NotImplementedError

logger.info(f"Best Accuracy: {max_acc:.4f}")
```

Finally, design function to plot the CNN training and validation loss in `utils.py`

- Plot Training and Validation Loss (5%)

- `plot()`

- Plot the training loss and validation loss of the CNN model over epochs.
 - Save the resulting plot as 'loss.png'.
 - Requirements:
 - xlabel: 'Epoch'
 - ylabel: 'Loss'
 - **Paste your 'loss.png' in the report**

- Experiments (5%)

- In your 'loss.png', when did overfitting happen? Can you list 2 methods to reduce the chance of overfitting and **implement them in your code**? Please take a screenshot of the modified section of the code and analyze if the method has improved accuracy and reduced loss.

Part 2: Decision Tree

First, define functions to extract features using ConvNet in `decision_tree.py`.

- Feature Extraction (5%)

- `get_features_and_labels()`

- Extract features and labels from a given dataloader using a trained ConvNet model.
 - Returns:
 - features (List): A list of feature maps extracted by the model.
 - labels (List): A list of the corresponding labels for the input images.

- `get_features_and_paths()`

- Extract features and file paths from a given dataloader using a trained ConvNet model.
- Returns:
 - features (List): A list of feature maps extracted by the model.
 - paths (List): A list of file paths corresponding to the images used for feature extraction.

Then, implement your decision tree model using **Information Gain** as the criterion to evaluate splits.

- Model Architecture (20%)

- **_build_tree()**
 - This function needs to handle the tree-building process recursively. Include stopping conditions, such as maximum depth, minimal data points, or reaching pure nodes
- **predict()**
 - This method should rely on **_predict_tree** to traverse the decision tree for each sample in the test dataset
 - Returns: A NumPy array of the predicted classes
- **_predict_tree()**
 - This recursive function should check whether **tree_node** is a leaf node. If it is, return the corresponding class label. If it is not, evaluate the splitting condition and traverse to the left or right child node accordingly.
- **_split_data()**
 - Implement logic for splitting the dataset into left and right subsets based on a specific feature and threshold.
 - Returns: the left and right subsets of features (**x**) and labels (**y**)
- **_best_split()**
 - Evaluate all possible splits for the current dataset and select the one that **maximizes information gain or minimizes entropy**.
 - Returns: the feature index and threshold that represent the best split.
- **_entropy()**
 - Calculate the entropy of the target labels (**y**) to measure the impurity.
 - Returns: entropy(float)

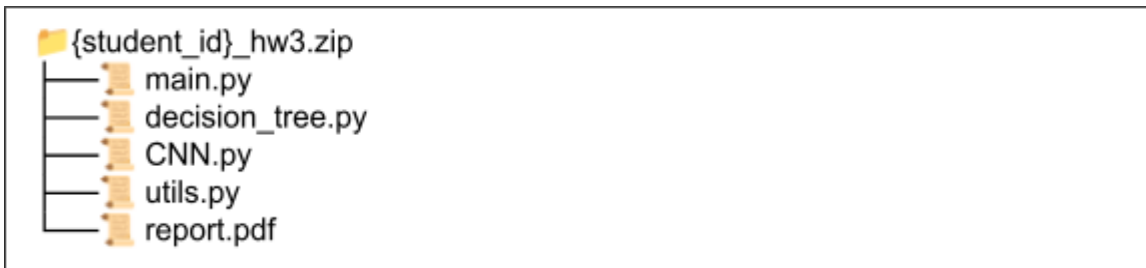
- Experiment (5%)

- In the template code, the `max_depth` is 7. How will the validation accuracy change when changing `max_depth` to 5 and 9? Why?

Submission

Due Date: 2025/04/21 17:00

Please compress your **source code, results, and report (.pdf)** into {STUDENT ID}_hw3.zip. The file structure should look like:



Please zip the contents, instead of the `_hw3` folder, so **there should NOT be a folder named `{student_id}_hw3` in the zip file.**

Do NOT zip the dataset and model checkpoints into submission file (-10 if not followed)

Requirements

1. Please modify the codes in **source code #TODO**
2. Do not import other packages.
3. The report should be written in **English**.
4. Please save the report as a **.pdf** file. (font size: 12)

Warnings

1. **Wrong submission format leads to -10 point.**
2. **No late submission is allowed.**

Kaggle Scoring (40%)

Your task in this section is to evaluate the accuracy of your CNN and Decision Tree models by submitting 'CNN.csv' and 'DecisionTree.csv' to Kaggle.

Part 1: CNN


- **Kaggle Evaluation Baseline (20%)**
 - Evaluate the accuracy in Kaggle (accuracy > 80% for full score)
 - Submission File:

For each filename in the test set, you must predict a class (0, 1, 2, 3, 4) for the label column. The file **should contain a header** and have the following format.

```
id,prediction
1,4
2,0
...
```

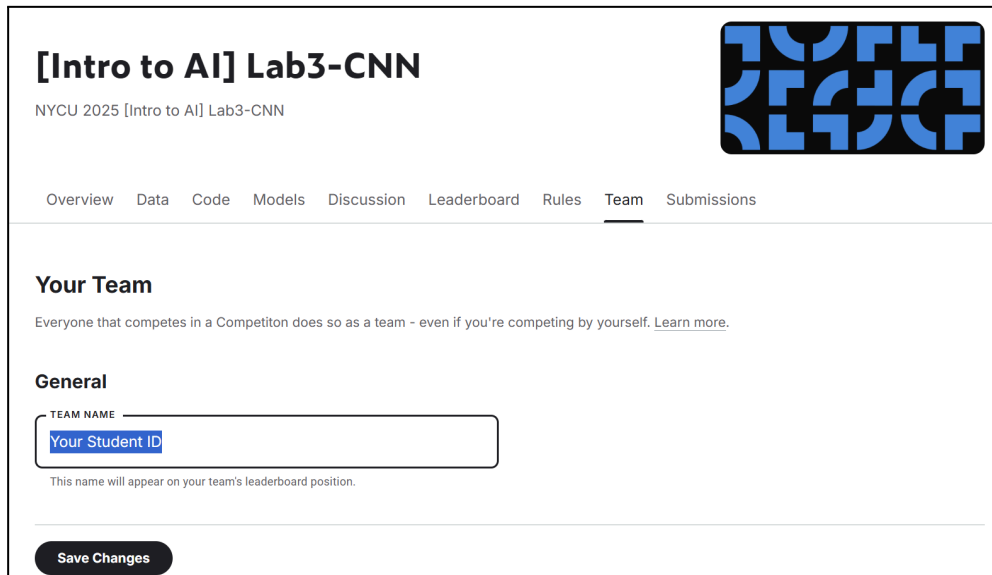
- Please include a screenshot of your final accuracy from the leaderboard in your report.

For example:

1	{your student ID}		0.8292	1	27m
---	-------------------	---	--------	---	-----

Don't forget to rename the team name to your [STUDENT_ID]

You can rename the team name at "Team"



- Submission Link
[\[Intro to AI\] Lab3-CNN | Kaggle](#)

Part 2: Decision Tree


- Kaggle Evaluation Baseline (20%)
 - Evaluate the accuracy in Kaggle (accuracy > 75% for full score)
 - Submission File:

For each filename in the test set, you must predict a class (0, 1, 2, 3, 4) for the label column. The file **should contain a header** and have the following format.

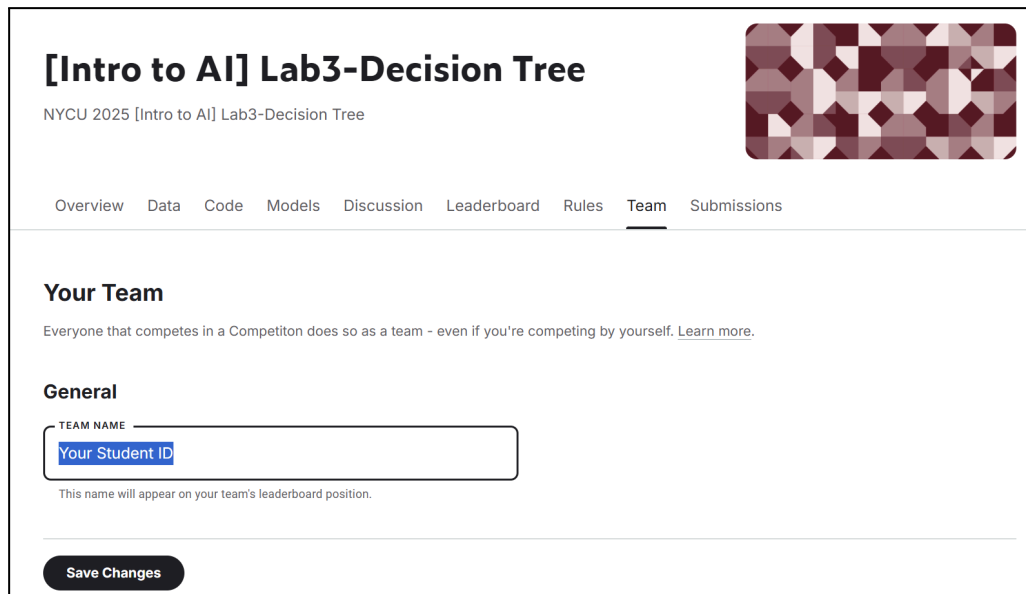
```
id,prediction
1,4
2,0
...
```

- Please include a screenshot of your final accuracy from the leaderboard in your report.

For example:

1	{your student ID}		0.8292	1	27m
---	-------------------	---	--------	---	-----

Don't forget to rename the team name to your [STUDENT_ID]
You can rename the team name at "Team"



[Intro to AI] Lab3-Decision Tree

NYCU 2025 [Intro to AI] Lab3-Decision Tree

Overview Data Code Models Discussion Leaderboard Rules **Team** Submissions

Your Team

Everyone that competes in a Competition does so as a team - even if you're competing by yourself. [Learn more.](#)

General

TEAM NAME

This name will appear on your team's leaderboard position.

Save Changes

- Submission Link
[\[Intro to AI\] Lab3-Decision Tree | Kaggle](#)

Grading rubrics

We will grade you according to your Kaggle accuracy:

- CNN.weak-baseline = 70 / CNN.strong-baseline = 80
- Decision-Tree.weak-baseline = 65/ Decision-Tree.strong-baseline = 75
- Less than weak-baseline (Acc < w.baseline): Score = 0
- Between weak-baseline and strong baseline:
 $(X - w.baseline) / (s.baseline - w.baseline) * 20$
- Achieve strong-baseline: Score = 20

CNN



Decision Tree



Requirements

1. Kaggle maximum daily submission limit: 10

2. Rename the team name to your [STUDENT_ID].

Warnings

1. No late submission is allowed.
2. Your Kaggle score will be 0 without renaming the team name to your [STUDENT_ID].

QA Page

If you have any questions about this homework, please ask them on **E3 forums**. We will answer them as soon as possible. Additionally, we encourage you to answer other students' questions if you have any idea 😊

No Plagiarism Allowed

Please remember that all submitted work must be entirely your own. Even if you use ChatGPT or other AI tools, your submission will be considered plagiarized if it is identical or highly similar to someone else's. Academic integrity is non-negotiable.

GPU Support

If you don't have a GPU or it's too slow, please use Google Colab. It provides a limited but enough FREE GPU.

<https://colab.google/>

Reference

[1] CNN Architecture Explained (Medium)

<https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243>

[2] Patterson, J., & Gibson, A. (2017). Deep learning: A practitioner's approach. O'Reilly Media.

[3] Nilsson, N. J. (n.d.). Introduction to Machine Learning: An Early Draft of a Proposed Textbook. Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA.

[4] Decision trees (GitHub pages)

<https://goldinlocks.github.io/Decision-trees-in-python/>

[5] Decision trees (GeeksforGeeks)

[Python | Decision tree implementation - GeeksforGeeks](#)

[6] Decision trees (Alice Gao)

[Lecture 7 Decision Trees](#)