ITAI 2377 Lab 04: Deep Learning Data Preprocessing

Instructor: [Oluchi Obinna] **Date:** [09/17/2025]

Introduction

Welcome to Lab 04! We'll explore the critical role of data preprocessing in deep learning. Even though models can extract features, preprocessing is essential for optimal performance. We'll cover various data types and apply preprocessing techniques. Resources in Google Colab are limited, so efficient coding is key!

Why Preprocess?

Why preprocess when models extract features?

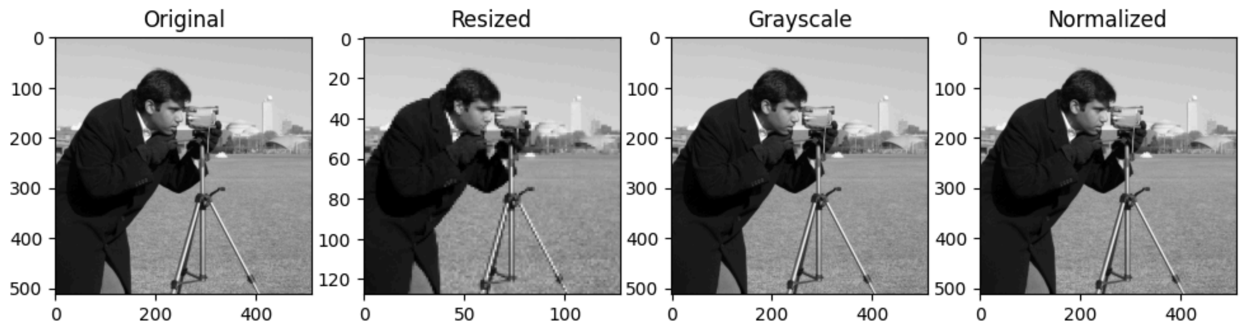
- **Standardization:** Models need consistent data formats and ranges.
- **Noise/Errors:** Raw data is messy. Preprocessing cleans it up.
- **Efficiency:** Cleaner data means faster training.
- **Results:** Good preprocessing helps models perform their best.

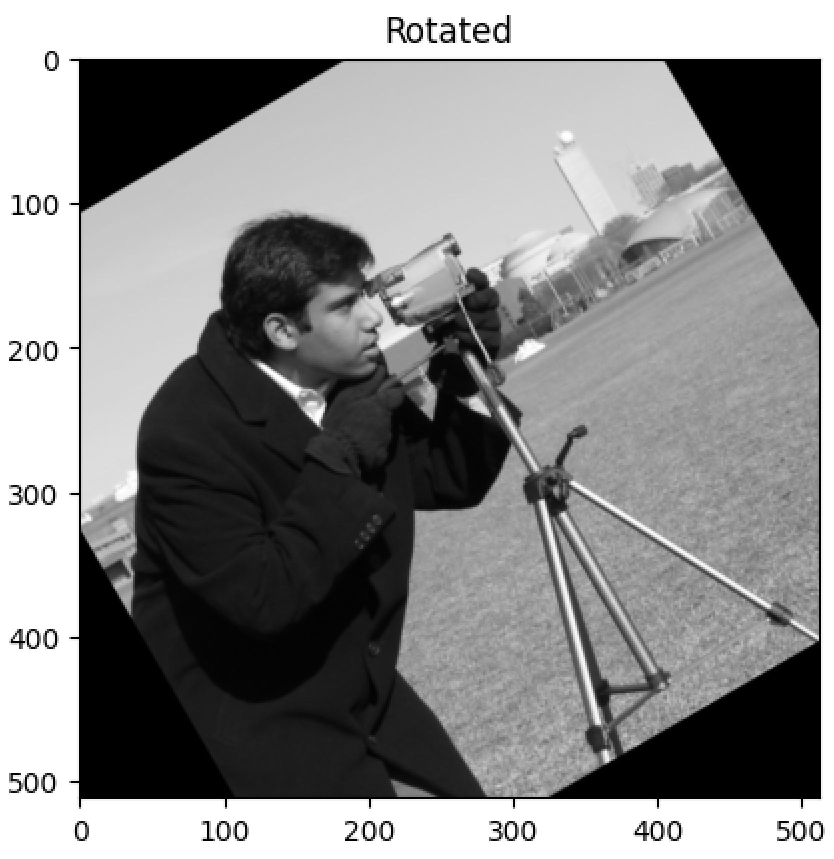
Think of preprocessing as a personal trainer for your model.

Data Types and Preprocessing Techniques

1. Image Data

In [2]:
```python
import cv2
import numpy as np
from skimage import data, img_as_float
import matplotlib.pyplot as plt

# Load a sample image (replace with your image path if you have one)
image = data.camera()  # Or use: image = cv2.imread("path/to/your/image.jpg")

# Resizing (Student Code: Resize the image to (128, 128))
# Hint: Use cv2.resize()
# YOUR CODE HERE
resized_image = cv2.resize(image, (128, 128))


# Color space conversion (to grayscale)
# The image is already grayscale, so this step is not needed.
# gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

# Normalization (pixel values 0-1)
normalized_image = img_as_float(image)

# Display images
plt.figure(figsize=(12, 6))

plt.subplot(1, 4, 1), plt.imshow(image, cmap='gray'), plt.title("Original")
plt.subplot(1, 4, 2), plt.imshow(resized_image, cmap='gray'), plt.title("Resiz
plt.subplot(1, 4, 3), plt.imshow(image, cmap='gray'), plt.title("Grayscale") #
plt.subplot(1, 4, 4), plt.imshow(normalized_image, cmap='gray'), plt.title("No
plt.show()

# Data Augmentation (rotation - Student Code: Rotate by 30 degrees)
# Hint: Use cv2.getRotationMatrix2D and cv2.warpAffine
angle = 30 #YOUR CODE HERE
rows, cols = image.shape[:2]
M = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, 1)
rotated_image = cv2.warpAffine(image, M, (cols, rows))

plt.imshow(rotated_image, cmap='gray'), plt.title("Rotated")
plt.show()
```





2. Text Data

In [4]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import string

nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
nltk.download('wordnet', quiet=True)
nltk.download('punkt_tab', quiet=True)

text = "This is a fun example sentence with stop words and punctuation!"

# Tokenization (lowercase)
tokens = word_tokenize(text.lower())

# Stop word removal (Student Code: Remove stop words and punctuation)
# Hint: Use the 'stop_words' set and list comprehension
stop_words = set(stopwords.words('english'))
# YOUR CODE HERE
filtered_tokens = [w for w in tokens if not w in stop_words and w not in strir


# Lemmatization (Student Code: Lemmatize the filtered tokens)
# Hint: Use WordNetLemmatizer()
lemmatizer = WordNetLemmatizer()
# YOUR CODE HERE
lemmatized_tokens = [lemmatizer.lemmatize(w) for w in filtered_tokens]


print("Original:", text)
print("Tokens:", tokens)
print("Filtered:", filtered_tokens)
print("Lemmatized:", lemmatized_tokens)
```

```
Original: This is a fun example sentence with stop words and punctuation!
Tokens: ['this', 'is', 'a', 'fun', 'example', 'sentence', 'with', 'stop', 'word
s', 'and', 'punctuation', '!']
Filtered: ['fun', 'example', 'sentence', 'stop', 'words', 'punctuation']
Lemmatized: ['fun', 'example', 'sentence', 'stop', 'word', 'punctuation']
```

3. Time Series Data

In [5]:

```
import pandas as pd
import numpy as np

# Sample data (with a missing value)
data = {'Date': pd.to_datetime(['2024-01-01', '2024-01-02', '2024-01-03', '202
        'Value': [10, 12, 15, np.nan, 18]}

df = pd.DataFrame(data)

# Missing value handling (Student Code: Use backward fill to fill missing valu
# Hint: Use fillna() with method='bfill'
# YOUR CODE HERE
df['Value'].fillna(method='bfill', inplace=True)

# Normalization (min-max scaling - Student Code: Normalize the 'Value' column)
# YOUR CODE HERE
min_val = df['Value'].min()
max_val = df['Value'].max()
df['Normalized'] = (df['Value'] - min_val) / (max_val - min_val)

print(df)
```

```
        Date  Value  Normalized
0 2024-01-01   10.0       0.000
1 2024-01-02   12.0       0.250
2 2024-01-03   15.0       0.625
3 2024-01-04   18.0       1.000
4 2024-01-05   18.0       1.000
```

```
/tmp/ipython-input-3069143717.py:13: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['Value'].fillna(method='bfill', inplace=True)
/tmp/ipython-input-3069143717.py:13: FutureWarning: Series.fillna with 'method'
is deprecated and will raise in a future version. Use obj.ffill() or obj.bfil
l() instead.
  df['Value'].fillna(method='bfill', inplace=True)
```

Tip: Explore different methods for handling missing values (e.g., backward fill, interpolation). Consider feature engineering techniques like creating lagged variables.

Questions (Markdown Cell)

1. Why is data preprocessing still important even with deep learning's feature extraction capabilities?

Even though deep learning models can find patterns, they still need clean and consistent data. Preprocessing removes errors, fills gaps, and makes training faster and more accurate.

2. Explain the difference between normalization and standardization. When would you choose one over the other?

Normalization: Puts values into a fixed range, usually 0–1. Good for things like image pixels. Standardization: Centers data around mean = 0 with standard deviation = 1. Useful when data looks more normal (bell-shaped). Choice: Normalize for bounded data like images; standardize for data that follows a normal distribution.

3. Describe a scenario where data augmentation would be particularly useful.

In medical imaging, there aren’t always many X-rays to train on. Augmentation (rotating, flipping, etc.) creates more training examples, helping the model learn better.

4. What are some potential challenges or pitfalls to avoid during data preprocessing? How can you mitigate them?

Over-cleaning: Might remove important data = be careful with thresholds.

Data leakage: Using test data during preprocessing = always split data first.

Bad missing value handling: Can bias results = pick the right fill method.

Slow processing: Too many steps waste time = keep it efficient.

5. Choose one of the data types covered in the lab (images, text, time series). Describe a specific real-world application that uses deep learning and explain how preprocessing would be crucial for that application.

Data Type: Text

Application: Sentiment analysis of product reviews.

Preprocessing: Removing stop words, punctuation, and lemmatizing words keeps the model from treating “happy,” “happier,” and “happiness” as totally separate, improving accuracy.

Deliverables

- **Completed Notebook (PDF):** This notebook with your code, outputs, and answers to the questions.
- **Reflective Journal:** A short journal (1-2 pages) reflecting on your learning experience in this lab. Consider the following prompts:
  - What did you learn in this lab?
  - What challenges did you encounter? How did you overcome them?
  - Were there any concepts that you already knew?
  - Did anything surprise you?
  - What are some potential real-world applications of the preprocessing techniques you learned?
  - What further learning or exploration would you like to pursue related to data preprocessing?

Remember to save your notebook with the outputs and convert it to PDF before submission. Good luck!

Reflective journal

In this lab, I learned how important preprocessing is before training a deep learning model. Even though these models are good at finding patterns, they still need clean and consistent data to work well. Preprocessing makes training faster, reduces errors, and often improves results. I also got hands-on practice with three types of data. For images, I resized, normalized, and rotated them. For text, I tokenized, removed stop words, and lemmatized words so the model would treat similar words the same. For time series, I worked with missing values and applied normalization so the data could be compared more fairly.

One challenge I faced was the warning message when I tried to fill missing values in the time series data. At first, it was confusing, but I learned the updated way to handle it by using the newer functions in pandas. Another challenge was remembering the correct syntax for rotating images in OpenCV. It took me a couple of tries, but with the lab hints and some trial and error, I got it working. These small struggles actually helped me understand the steps more clearly.

Some of the concepts were familiar to me already. For example, I had some knowledge of text preprocessing like tokenization and stop word removal. However, lemmatization was new, and I realized how useful it is for reducing words to their base form. I also had a general idea of normalization and standardization, but this lab helped me see when each should be used and why they matter.

What surprised me most was how much impact small preprocessing steps can have. Something as simple as resizing an image or normalizing pixel values makes the data much easier for the model to learn from. The same goes for text: combining words like “words” and “word” into a single form can really reduce complexity and improve performance.

The techniques from this lab have clear real-world applications.For ex, Image preprocessing is used in medical imaging and self-driving cars. Text preprocessing is needed for chatbots, review analysis, and translation systems. Time series preprocessing helps with stock predictions, weather forecasting, and sensor data analysis.

Completing this lab showed me that preprocessing is not just a boring first step, but a powerful way to make deep learning models more effective. Going forward, I want to explore more advanced data augmentation techniques, like generating new images with AI, and also learn how preprocessing is handled at scale in real-world AI systems. That way, I can see how these concepts connect to real projects beyond the lab.

Sources

1. https://towardsdatascience.com/text-normalization-with-spacy-and-nltk-1302ff430119