ITAI 2377 Lab 04: Deep Learning Data Preprocessing

Instructor: [Your Name] **Date:** [Date]

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?

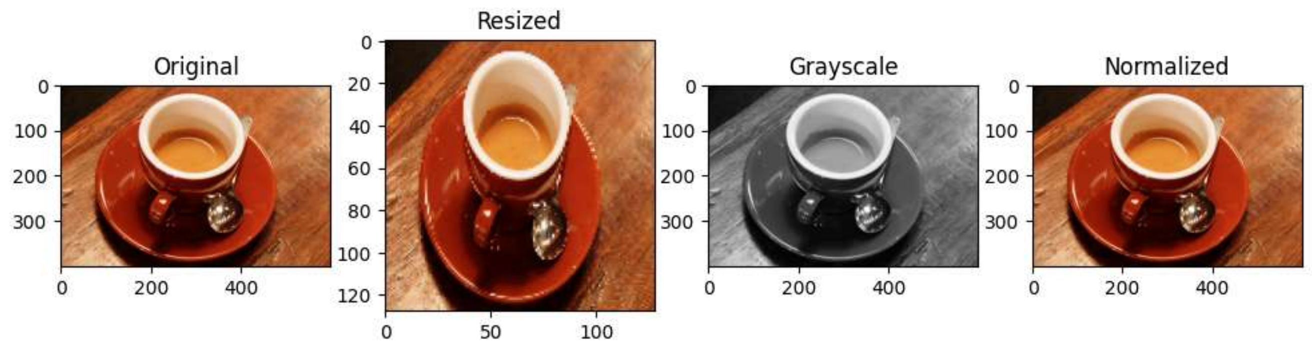
- **Standardization:** Models need consistent data formats and ranges.
- **Noise/Errors:** Raw data is messy. Preprocessing cleans it up.
- **Efficiency:** Cleaner data means faster training.
- **Results:** Good preprocessing helps models perform their best.

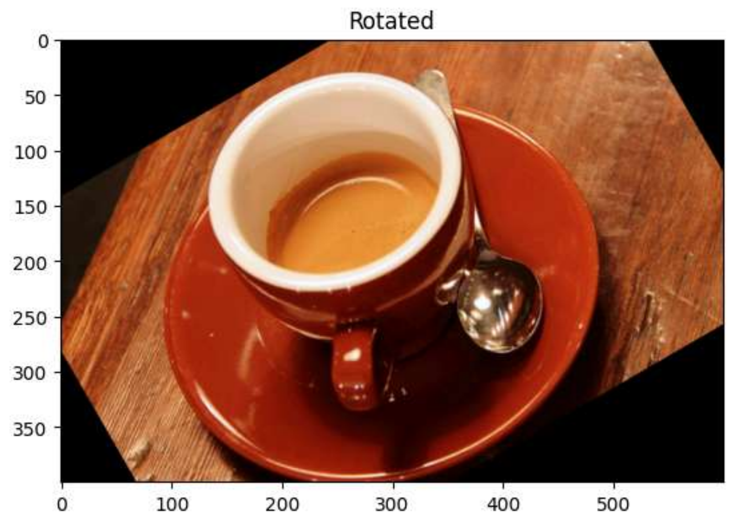
Think of preprocessing as a personal trainer for your model.

Data Types and Preprocessing Techniques

1. Image Data

```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.coffee()  # I used an image included in skimage.data

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

# Color space conversion (to grayscale)
# It appears original photo was already grayscale
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), plt.title("Original")
plt.subplot(1, 4, 2), plt.imshow(resized_image), plt.title("Resized")
plt.subplot(1, 4, 3), plt.imshow(gray_image, cmap='gray'), plt.title
("Grayscale")
plt.subplot(1, 4, 4), plt.imshow(normalized_image), plt.title("Normalized")
plt.show()

# Data Augmentation (rotation - Student Code: Rotate by 30 degrees)
# Hint: Use cv2.getRotationMatrix2D and cv2.warpAffine
angle = 30
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), plt.title("Rotated")
plt.show()
```





2. Text Data

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import string

nltk.download('punkt_tab', quiet=True) # The errors I got kept pointing to a 'punkt_tab' location but the original code was just pointin
nltk.download('stopwords', quiet=True)
nltk.download('wordnet', 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'))
filtered_tokens = [w for w in tokens if not w in stop_words and w not in string.punctuation]


# Lemmatization (Student Code: Lemmatize the filtered tokens)

lemmatizer = WordNetLemmatizer()

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', 'words', 'and', 'punctuation', '!']
Filtered: ['fun', 'example', 'sentence', 'stop', 'words', 'punctuation']
Lemmatized: ['fun', 'example', 'sentence', 'stop', 'word', 'punctuation']
```

3. Time Series Data

```
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', '2024-01-04', '2024-01-05']),
        'Value': [10, 12, 15, np.nan, 18]}

df = pd.DataFrame(data)

# Missing value handling (Student Code: Use backward fill to fill missing values)
# Hint: Use fillna() with method='bfill'

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-9-3069143717.py:13: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
```

```
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Value'].fillna(method='bfill', inplace=True)
/tmp/ipython-input-9-3069143717.py:13: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use
  df['Value'].fillna(method='bfill', inplace=True)
```

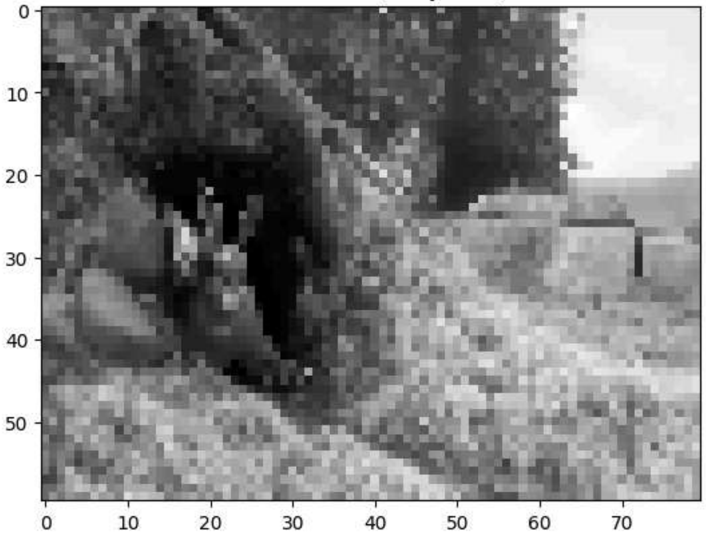
4. Optional: Video Data (Simplified)

```python
from google.colab import files
import cv2
import matplotlib.pyplot as plt
from pathlib import Path

video_filename = next(iter(uploaded))  # ← This happens BEFORE upload
cap = cv2.VideoCapture(str(video_path))  # ← Uses video_path, which isn't defined yet

# Upload a file interactively
uploaded = files.upload()
video_path = Path(next(iter(uploaded)))
cap = cv2.VideoCapture(str(video_path))

# I had to go this route with the code above, I got everything mixed up. I still have issues on when to know when to use windows directc

if not cap.isOpened():
    print(f"❌ Couldn't open video: {video_path.name}")
else:
    print(f"✅ Opened video: {video_path.name}")
    if ret:
        # Resize the frame (for faster processing - Student Code: Resize to (80, 60))
        resized_frame = cv2.resize(frame, (80, 60))

        # Convert to grayscale
        gray_frame = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2GRAY)  # OpenCV uses BGR

        # Display the frame (optional)
        plt.imshow(gray_frame, cmap='gray')
        plt.title("Video Frame (Grayscale)")
        plt.show()

    cap.release()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving SampleVideo.mp4 to SampleVideo (8).mp4
✅ Opened video: SampleVideo (8).mp4
```

Video Frame (Grayscale)

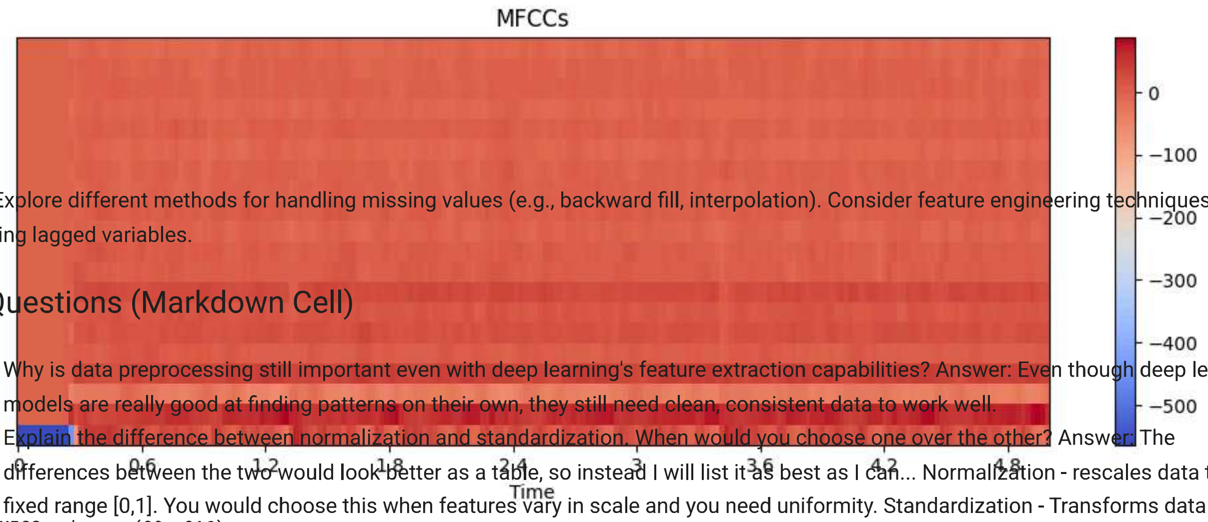


5. Optional: Audio Data (Simplified)

```python
import librosa
import numpy as np
import matplotlib.pyplot as plt

from google.colab import files
uploaded = files.upload()
audio_path = next(iter(uploaded))

# Load an audio file (replace with your audio path)
# Replace with your audio file path (or upload one to Colab)
y, sr = librosa.load(audio_path, duration=5)  # Load a maximum of 5 seconds

# Feature extraction (MFCCs - Student Code: Extract 20 MFCCs)
# Hint: Use librosa.feature.mfcc() with n_mfcc=20
# YOUR CODE HERE
mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=20)

# Display MFCCs (optional)
plt.figure(figsize=(10, 4))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
plt.colorbar()
plt.title('MFCCs')
plt.tight_layout()
plt.show()

# Normalize MFCCs (example)
mfccs_normalized = (mfccs - np.mean(mfccs)) / np.std(mfccs)

print("MFCCs shape:", mfccs.shape)
print("Normalized MFCCs shape:", mfccs_normalized.shape)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 03.Bleed (2023, 15Th Anniversary Remastered Edition).mp3 to 03.Bleed (2023, 15Th Anniversary Remastered Edition).mp3
```



```
MFCCs shape: (20, 216)
Normalized MFCCs shape: (20, 216)
```

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? Answer: Even though deep learning models are really good at finding patterns on their own, they still need clean, consistent data to work well.
2. Explain the difference between normalization and standardization. When would you choose one over the other? Answer: The differences between the two would look better as a table, so instead I will list it as best as I can... Normalization - rescales data to a fixed range [0,1]. You would choose this when features vary in scale and you need uniformity. Standardization - Transforms data to have a mean of 0 and standard deviation of 1. Use this when data may contain outliers or is not naturally bounded.
3. Describe a scenario where data augmentation would be particularly useful. Answer: A machine learning model for people who are deaf or hard of hearing. A computer vision model that is able to recognize sign language from video and converts it to text in real-time. Augmentation helps in this space without having to record thousands of videos. Apply small rotations or flips to reflect differnt hand orientations. Background substitution (or blurring) to help the model focus on hands, not surroundings.
4. What are some potential challenges or pitfalls to avoid during data preprocessing? How can you mitigate them? Answer: Some