Facial Keypoint Detection

Image Processing using Machine Learning

Visualizing Data

- By using Matplotlib we can make a plot with the raw image data and use the keypoint coordinates to plot the key points on the subjects face to help us visualize the points we're looking for.

- This can be especially helpful when trying to actually visualize why some photos, such as those from the "Unclean Train Data" are missing key points.

- It can also be used to visualize the Data Augmentation options as we will see in later slides.

```python
def plot_sample(image, keypoint, axis, title):
    image = image.reshape(96,96)
    axis.imshow(image, cmap='gray')
    axis.scatter(keypoint[0::2], keypoint[1::2], marker='x', s=20)
    plt.title(title)
```

Data Preprocessing

- All data in this project is loaded into our program using the Pandas python library.
- Pandas is a software library for data manipulation and analysis .
- Pandas is best used when manipulating data tables such as the ones we use for graphing the position of facial features.
- Using this library we gain access to many functions that are useful for loading and organizing our data into smart objects known as Dataframes.
- Once our data is loaded into a Pandas dataframe we can begin visualizing in many different ways.
- This includes data merging, filtering, re-indexing or creating subsets of the data.
- For the process of exploratory analysis you can inspect different columns, rows and even individual cells.
- Although it is not useful for changing the results of our training for this project, you can also use the groupby() method to sort through data that you want to observe or potentially segregate from the rest of the dataset

Data Preprocessing

- The other important library we utilized in our research was the Scikit-learn library.
- The sklearn library is probably one of the most useful libraries for machine learning in Python. It contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
- For our purposes, we use sklearn for normalization of data, splitting our data into training, test and validation sets as well as filling in our missing data with SimpleImputer.
- We utilize a Train - 70% and Test - 30% split, then we take the train set and use K-fold from the Cross-validation package in Scikit-learn to create our validation set.
- K-fold splits the training data into K different partitions and cycles through which partition will be used as the validation set to allow the model to potentially be trained on a better portion of the training data.
- The other area where sklearn is useful is in the normalization of our data.
- Normalization is the process of scaling individual samples to have unit norm (i.e. puts it in the interval [0,1]). This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.
- This can also reduce the “importance” that may be given to some features with numbers on a larger scale when compared to those with smaller values and variances.
- We utilized 4 different scalars. StandardScalar(), MinMaxScaler(), MaxAbsScaler, and RobustScaler()

Handling the Missing Data

```
left_eye_center_x               10
left_eye_center_y               10
right_eye_center_x              13
right_eye_center_y              13
left_eye_inner_corner_x       4778
left_eye_inner_corner_y       4778
left_eye_outer_corner_x       4782
left_eye_outer_corner_y       4782
right_eye_inner_corner_x      4781
right_eye_inner_corner_y      4781
right_eye_outer_corner_x      4781
right_eye_outer_corner_y      4781
left_eyebrow_inner_end_x      4779
left_eyebrow_inner_end_y      4779
left_eyebrow_outer_end_x      4824
left_eyebrow_outer_end_y      4824
right_eyebrow_inner_end_x     4779
right_eyebrow_inner_end_y     4779
right_eyebrow_outer_end_x     4813
right_eyebrow_outer_end_y     4813
nose_tip_x                       0
nose_tip_y                       0
mouth_left_corner_x           4780
mouth_left_corner_y           4780
mouth_right_corner_x          4779
mouth_right_corner_y          4779
mouth_center_top_lip_x        4774
mouth_center_top_lip_y        4774
mouth_center_bottom_lip_x       33
mouth_center_bottom_lip_y       33
Image                            0
dtype: int64
```

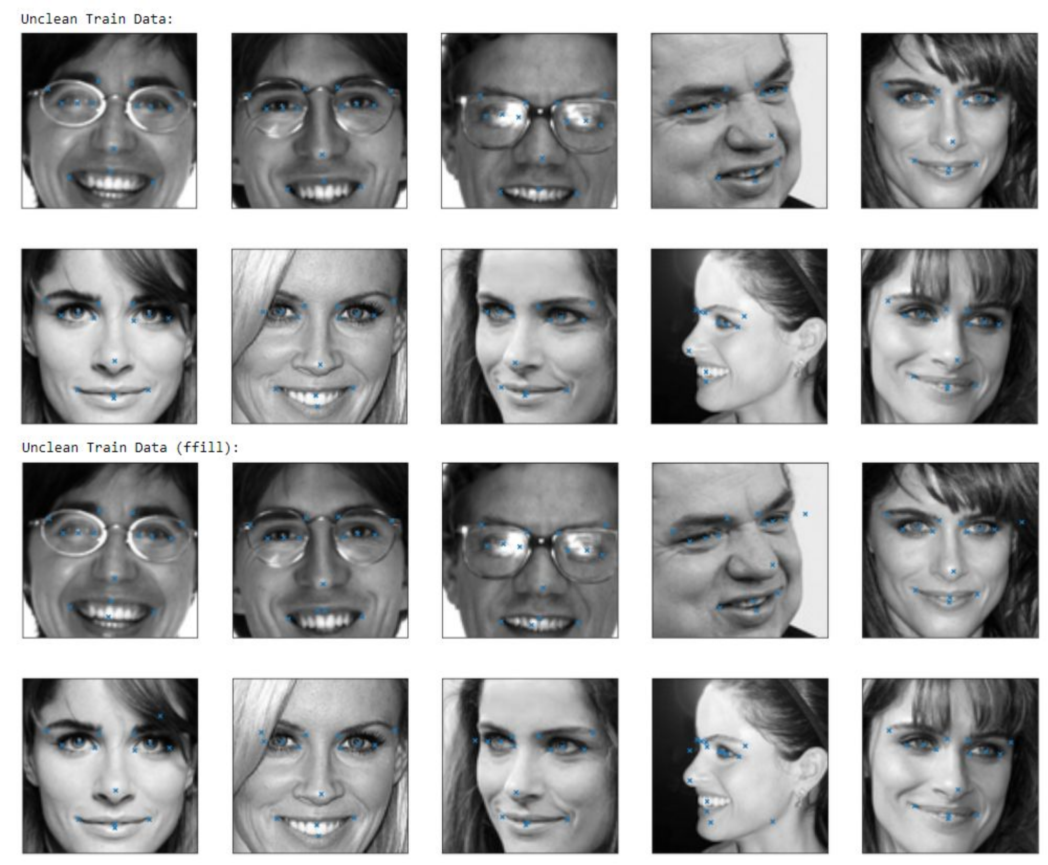
- By utilizing the Dataframe.isnull() method from pandas, we can determine how many of our keypoints are missing by summing up all the missing values for each key point.
- Considering the dataset contains 7049 images, there is roughly 68% of the key points for data missing in various samples.
- There are generally two methods that are utilized to fix this problem.
- You can discard the rows that contain incomplete observations, but then you would lose over half of the dataset and you would also lose a lot of valuable information in the points that each dropped image would have had.
- Alternatively, you could fill in the missing data using methods such as: ffill, bfill, mean, interpolation, etc.
- The method you choose for extrapolating the missing data should depend solely on the type of data set you are working with.

Handling the Missing Data

```
clean_train_data = train_data.dropna()
print("clean_train_data shape: {}".format(np.shape(clean_train_data)))

unclean_train_data = train_data.fillna(method = 'ffill')
print("unclean_train_data shape: {}\n".format(np.shape(unclean_train_data)))
```

- By utilizing the Pandas library you can see the two functions are simple for handling the missing data.
- The top function will process the whole dataset, remove rows with missing or NaN values, then return the dataset with only completely filled out rows of data. Since we did not set the “inPlace” flag to true the original train_data dataframe remains unaffected and serves to only pass off the data into a new variable.
- Thus, when we create the unclean_train_data dataframe the fillna() method will once again process the whole dataset and fill in the missing or NaN values based on the method selected.
- The methods available for extrapolation include: ‘backfill’/‘bfill’, ‘pad’/’ffill’, None
- pad / ffill: propagate last valid observation forward to next valid
- backfill / bfill: use next valid observation to fill gap.
- It is also possible to pass in a list of values or a single value to use for the missing data in a particular column (Which is how you could fill in the data using the mean value of a column).

Results of Filling in Data



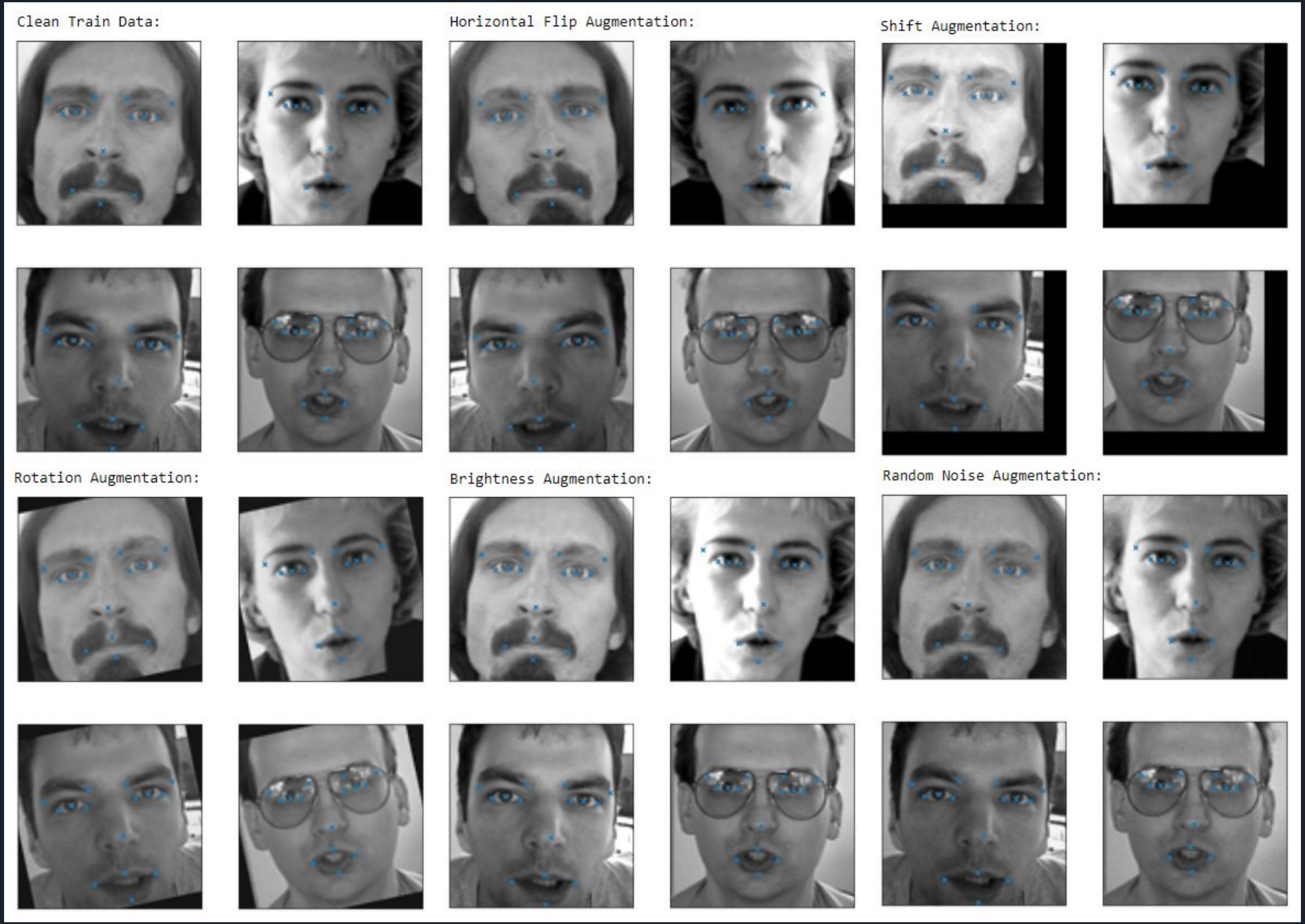
Data Augmentation

- Data manipulation enables the model to perform consistently throughout the whole dataset by overseeing invariances in the images within the training dataset.
- For instance, Geometric transformations help in our case with adding extra data for training by changing the spatial positions of the subjects faces in the frame.

```
horizontal_flip = True
rotation_augmentation = True
brightness_augmentation = True
shift_augmentation = True
random_noise_augmentation = True

include_unclean_data = True    # Whether to include samples with missing keypoint values. Note that the missing
                               #values would however be filled using Pandas' 'ffill' later.

sample_image_index = 20        # Index of sample train image used for visualizing various augmentations

rotation_angles = [12]    # Rotation angle in degrees (includes both clockwise & anti-clockwise rotations)
pixel_shifts = [12]    # Horizontal & vertical shift amount in pixels (includes shift from all 4 corners)

NUM_EPOCHS = 80
BATCH_SIZE = 64
```

Boolean Flag Variables for current Augmentation Choices

Geometric transformations - Flipping, cropping, rotations, translational shifts	Model overcomes positional biases in the dataset.
Color space transformations - Brightness, contrast, saturation	Model performs well when lighting and colors vary greatly from the rest of the dataset.
Kernel filters - Sharpening or blurring an image	Performance increase when predicting outputs of blurry or low-quality images.


Clean Train Data:
Horizontal Flip Augmentation:
Shift Augmentation:
Rotation Augmentation:
Brightness Augmentation:
Random Noise Augmentation:

Thank You For Your Time!