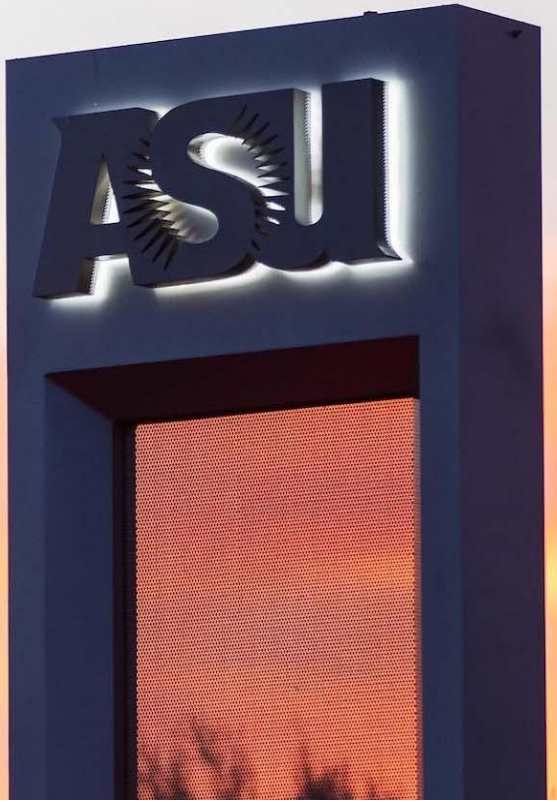
PROJECT PORTFOLIO



Arizona State University

- Harkirat Singh Chahal
- MS Robotics & Autonomous Systems (Systems Engg)

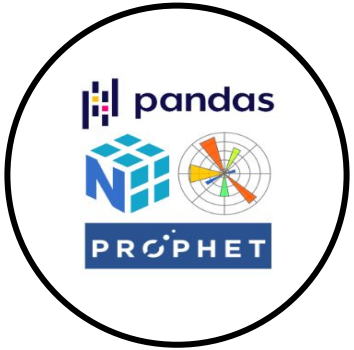
About Me

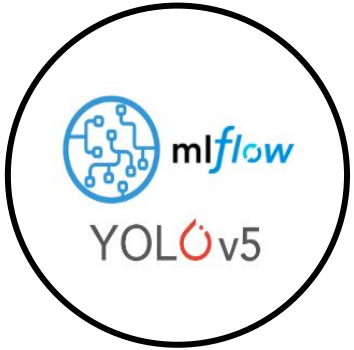
- **Name:** Harkirat Singh Chahal
- **Current Role:** Graduate Student in Robotics & Autonomous Systems at ASU
- **Interest Areas:** Robotics, Machine Learning, Data Science
- **Professional Experience:**
  - Data Scientist at Mamaearth, India's leading D2C FMCG Company (2 years)
  - Data Scientist at GEODIS Logistics LLC
- **Highlight Courses at ASU:**
  - Artificial Neural Computation
  - Experimentation & Deployment of Robotic Systems
  - Embedded Machine Learning
  - Robotics 1

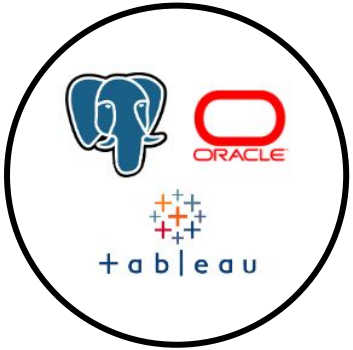
Research Interests

- **Predictive Modeling:** Utilizing labeled datasets to develop models, forecasting future outcomes and trends.
- **Data Management:** Ensuring optimal data quality through rigorous cleaning and preprocessing methodologies.
- **In-depth Statistical Analysis:** Employing statistical methods to decipher patterns, correlations, and insights within the data.
- **Time Series Exploration:** Conducting a thorough analysis of time-sequenced data to enhance forecasting accuracy.
- **Computer Vision:** Implementing ML models to interpret and make decisions based on visual data.
- **Robotics & ROS Development:** Engaging in the development and enhancement of robotic systems and Robot Operating System (ROS) applications.

Technical Skills







Programming & Tools

- **Languages:** Python, Matlab
- **Robotics & Simulation:** ROS, Simulink
- **Additional:** GIT, Bash Scripting, Docker, MS Excel (Advanced)

Python Libraries

- **Data & ML:** Pandas, NumPy, Sklearn, PyTorch, TensorFlow
- **Visualization & Time Series:** Matplotlib, Seaborn, Prophet
- **ETL & Management:** PySpark, Apache Airflow, mlflow

ML & AI

- **Applied Techniques:** Regression, SVM, KNN, Decision Trees
- **Neural Networks:** CNN, RNN
- **Forecasting:** Time Series Analysis, ARIMA, fbProphet
- **Model Management:** MLflow, evidently
- **Computer Vision:** OpenCV, YOLO V5

Databases & Dashboards

PostgreSQL, MySQL, Snowflake, Oracle, PowerBi, Tableau

My Projects

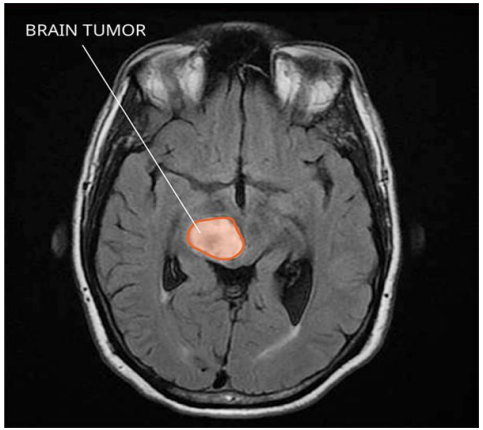
- Medical Image Segmentation using Transformers Model - Trans tsv2
- RSNA Breast Cancer Detection
- Integrating Deep Learning for Autonomous Path Tracking in Robotics
- Summer 2023 Internship Project
- Music Generation using LSTM

Medical Image Segmentation using Transformers Model TransBTSV2

EEE 511 Artificial Neural Computation

Project Scope

- **Project Objective:** Implement the TransBTSV2 image segmentation algorithm on 3D medical images to enhance physicians' interpretations of MRI and CT scans.
- **Algorithmic Approach:** Employ Transformers, capitalizing on their proven ability to retain long-range information and capture global features, to streamline the segmentation process.
- **Neural Network Strategy:** Prioritize a broad neural network (NN) architecture over depth, ensuring efficient and effective image processing.
- **Incorporation of Local Features:** Leverage Convolutional Neural Networks (CNNs) to effectively manage and process local features, which are pivotal for dense 3D image segmentation tasks.
- **Validation & Collaboration:** Utilize metrics such as Dice scores for validation, submitting output to CBICA (University of Pennsylvania) to ensure accurate and reliable results.

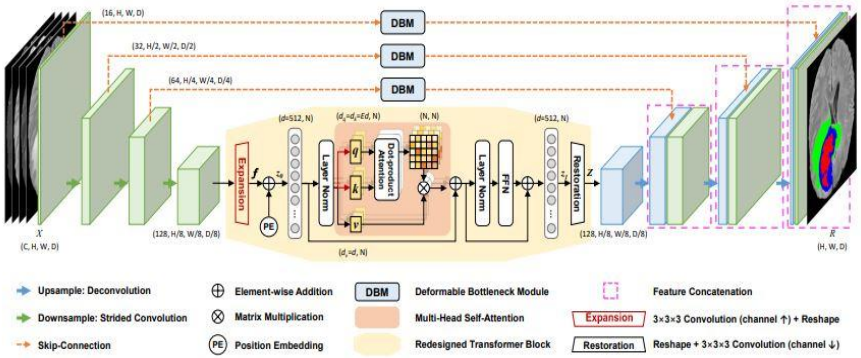


MRI Image

Continued..

Training the model & Results

- **Efficient Data Management:** Achieved 95% data reduction with bzip2, enabling real-time decompression during training.
- **Adaptable Platform Training:** Conducted model training across multiple platforms, including Google Colab and Agave Cluster (ASU Super Computer).
- **Strategic Warm-Up Period:** Utilized a warm-up strategy, gradually increasing from a minimized to the initial learning rate.
- **Data Augmentation Techniques:** Employed augmentation (random cropping, mirror flipping, intensity shifts) to prevent overfitting.
- **Rigorous Model Training:** Trained the model meticulously over 1,000 epochs to ensure comprehensive learning.



TransBTSV2 model followed for the application

Result

Successful Testing Across Two Datasets with TransBTS2 Model

- LiTS 2017
- kiTS 2019

Progressive Tumor Highlighting in Output Images

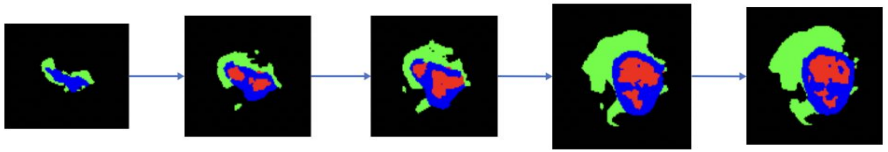
- Initial outputs presented completely black images.
- Tumor visibility progressively increased from the 1st to the 159th image.
- Optimal tumor delineation observed between the 75th and 80th images.

Method	Dice Score per Case(%)		Dice Score Global(%)	
	Lesion	Liver	Lesion	Liver
TransBTSV2	61.2	73.1	86.2	86.6

Performance Comparison On LiTS 2017 Testing Set.

Method	Kidney Dice (%)	Tumor Dice (%)	Composite Dice (%)
TransBTSV2	85.37	71.69	78.53

Performance Comparison On KiTS 2019 Testing Set.



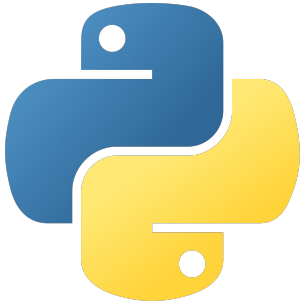
Tumor segmentation result using Transformer model

Tech Stack Used

PyTorch

NiBabel

Access a cacophony of neuro-imaging file formats



ASU Research Computing

ARIZONA STATE UNIVERSITY

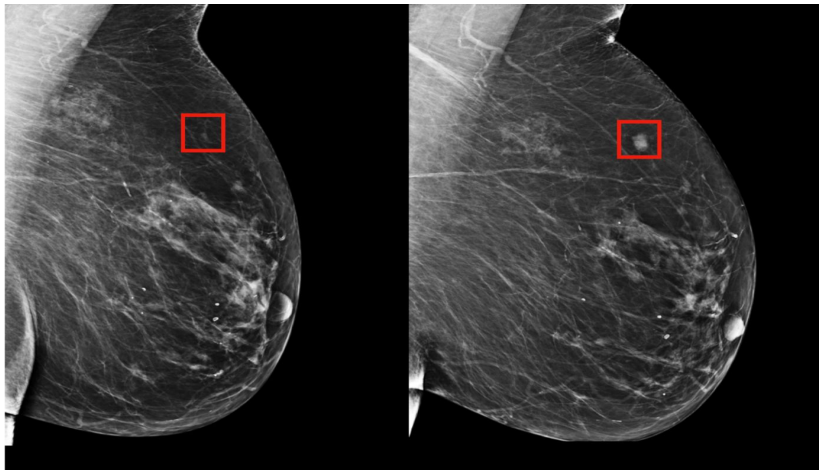
RSNA Breast Cancer Detection

Research Aide - WP Carey School of Business

Github: https://github.com/iamharkirat/breast_cancer.git

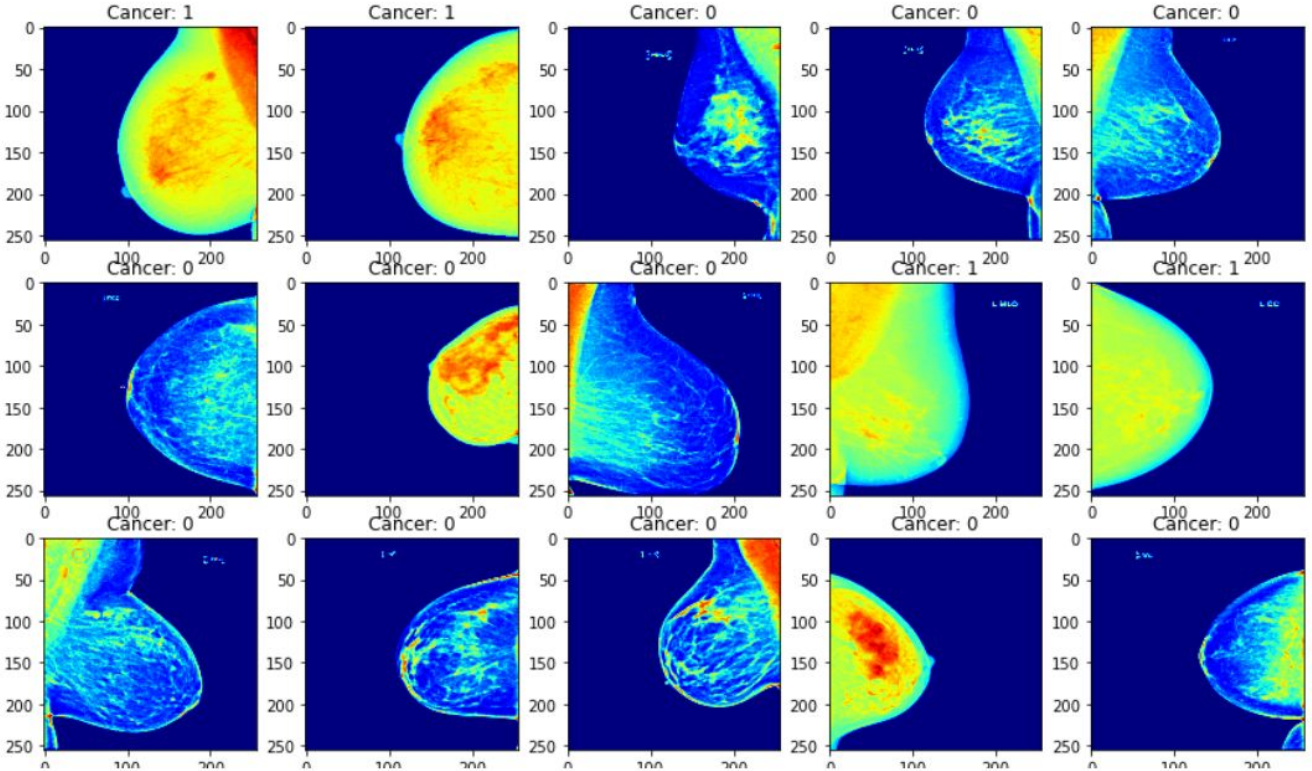
Project Scope

- **Primary Goal:** Develop a model to accurately identify breast cancer using screening mammograms, thereby enhancing the efficiency and precision of radiologists.
- **Challenges in Current Detection Methods:**
  - Dependency on highly-trained radiologists, making the screening process expensive.
  - High incidence of false positives, leading to unnecessary stress and additional medical procedures for patients.
- **Impact of Automation through ML:**
  - Facilitate early detection and treatment, crucial for reducing cancer fatalities.
  - Potentially streamline radiologists' evaluation process of screening mammograms.
- **Potential Outcomes:**
  - Enhance the quality and safety of patient care by improving detection automation.
  - Possibly reduce costs and curtail unnecessary medical procedures.



Mammogram Example

Sample set



Sample Random Images

```
majority_class_df=train_df[train_df['cancer'] == 0].sample(50)
minority_class_df=train_df[train_df['cancer'] == 1].sample(100)

final_df = pd.concat([majority_class_df, minority_class_df])
final_df=final_df.reset_index(drop=True)
final_df.head()
```

- Experiments were conducted to determine the optimal split of the data.
- The following options were evaluated:
  - Balanced dataset with equal number of healthy (500) and cancer (500) patients
  - Unbalanced dataset with fewer healthy patients (50) and more cancer patients (100)
  - Unbalanced dataset with more healthy patients (200) and fewer cancer patients (400)

➢ The unbalanced dataset worked better, and gave better validation accuracy

Normalize the images

After splitting the images into 0: healthy & 1: cancer sub-folders, we normalize the images to ensure that the input data is in a standardized format and scale.

Next we did some data integrity checks before passing the images to our model.

- Checked the min & max values of images
- Checked shapes of images (2 channel (224x224) grayscale image)

```
Cancer images check ===========
Image normalized: 0 1 (224, 224) /kaggle/working/train_images/cancer/60653_2052987229.png
Image normalized: 0 1 (224, 224) /kaggle/working/train_images/cancer/64439_84747386.png
Image normalized: 0 1 (224, 224) /kaggle/working/train_images/cancer/28989_1880776532.png
Image normalized: 0 1 (224, 224) /kaggle/working/train_images/cancer/7053_888903661.png
Image normalized: 0 1 (224, 224) /kaggle/working/train_images/cancer/38311_300211801.png
Image normalized: 0 1 (224, 224) /kaggle/working/train_images/cancer/31582_435931040.png
```

Data Augmentation

```
majority_class_df=train_df[train_df['cancer'] == 0].sample(50)
minority_class_df=train_df[train_df['cancer'] == 1].sample(100)

final_df = pd.concat([majority_class_df, minority_class_df])
final_df=final_df.reset_index(drop=True)
final_df.head()
```

- Experiments were conducted to determine the optimal split of the data.
- The following options were evaluated:
  - Balanced dataset with equal number of healthy (500) and cancer (500) patients
  - Unbalanced dataset with fewer healthy patients (50) and more cancer patients (100)
  - Unbalanced dataset with more healthy patients (200) and fewer cancer patients (400)

➢ The unbalanced dataset worked better, and gave better validation accuracy

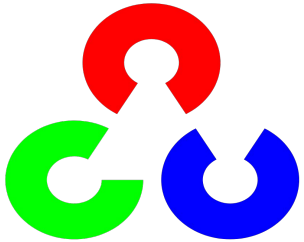
Image Pre-Processing Flow & Results

For this model, we kept the image in the original 3 channel rgb format, and applied 3 filters i.e Gamma, CLAHE 1 & CLAHE 2

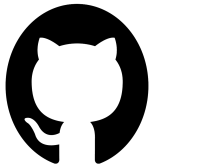
Thanks to the image pre-processing steps and augmentations, I was able to get an **accuracy of 84%** on the test dataset, an improvement from the previous 63% when I joined the team.

Tech Stack Used

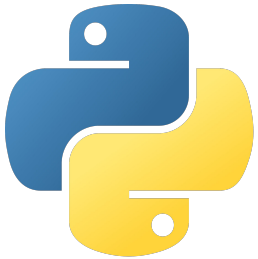
TensorFlow



OpenCV



GitHub



Integrating Deep Learning for Autonomous Path Tracking in Robotics

EGR 598 Ecperimentation & Deployment of Robotic Systems

Github: https://github.com/iamharkirat/EGR_598.git

Project Scope

Key Focus Areas:

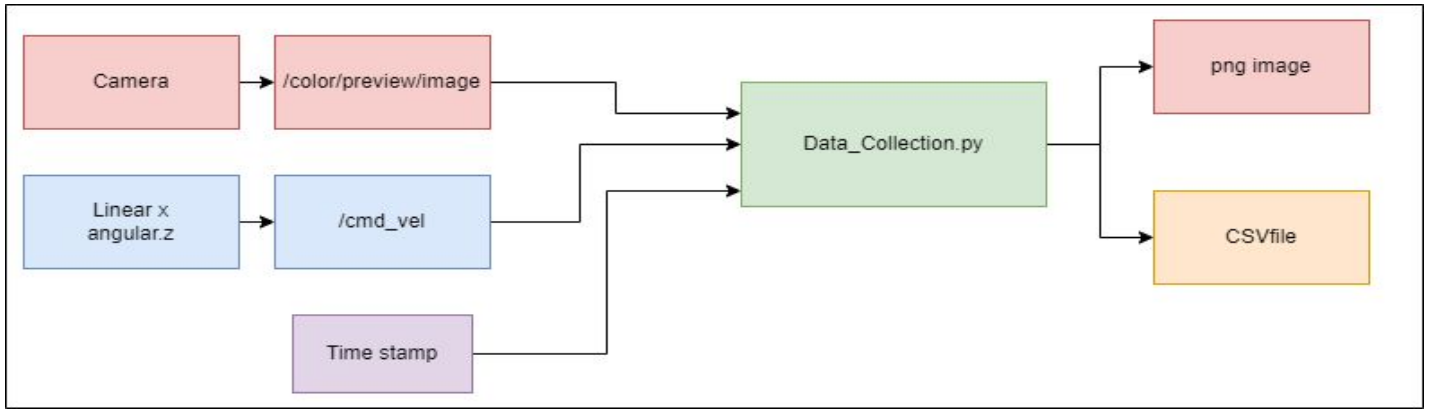
- Deep Learning & Computer Vision: Engage in an in-depth exploration and application of neural network architectures in robotics.
- Systems Integration: Gain and enhance knowledge in ROS and systems integration, central to implementing autonomous navigation.

Teamwork and Individual Contribution:

- Team Size: Collaborated within a 3-member team, each contributing specialized skills to the project.
- My Responsibilities:
  - Ensured a clean dataset, priming it for efficient training and validation.
  - Coding the Neural Network model, enabling the autonomous driving capabilities of the bot.

Turtlebot

Data Collection



Data Collection Process

- Utilized Teleop twist keyboard for turtle bot navigation on the track, with data from the oak-d camera published to /color/preview/image.
- Created a ROS2 node to subscribe to /color/preview/image and /cmd_vel topics, storing received data locally.
- Employed the DataCollectionNode class to manage data collection and storage, initializing subscribers and updating velocities.
- Converted image data to numpy arrays, saved as PNG files, and stored related data (timestamp, path, velocities) in a CSV for training labeling.

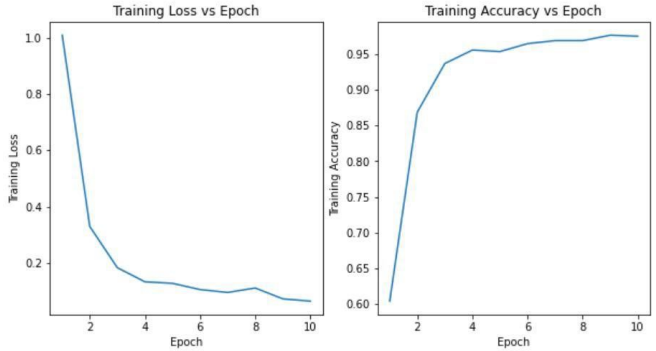
CNN Model

- Utilized the preprocess_image function to load and normalize image data into a NumPy array.
- Employed process_images_parallel to preprocess multiple images in parallel, optimizing performance.
- Created an array of preprocessed images and preprocessed labels through one-hot encoding.
- Split data into training and validation sets using train_test_split.
- Defined and compiled the model using Keras' Sequential API, incorporating convolutional, pooling, and dense layers.
- Trained the model with a batch size of 32 across 10 epochs, using the validation set to monitor performance.

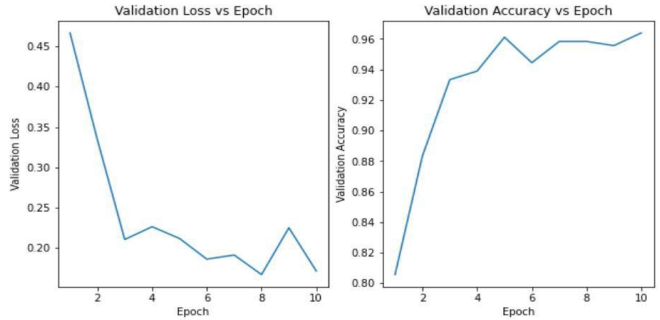
```
# Preprocess the labels
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(sampled_data["angular_z"])
labels = to_categorical(labels)

# Split the data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(images, labels, test_size=0.2, random_state=42)

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')  # We have 3 distinct labels
])

model.summary()

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, batch_size=32, epochs=10, validation_data=(X_val, y_val))
```

CNN Model

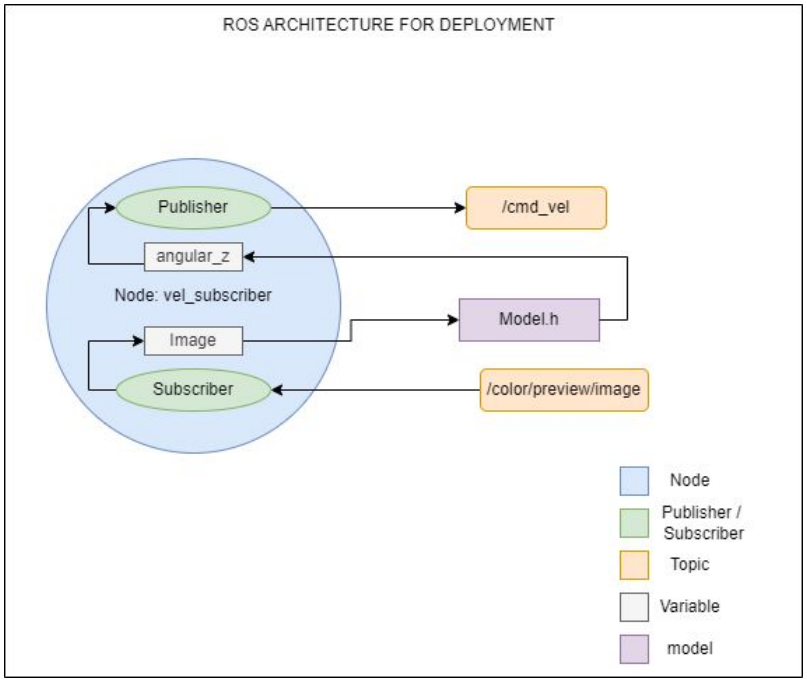
Model Evaluation



Custom model training metrics



Custom model validation metrics



ROS Architecture for Data Collection

Turtlebot in action…

Tech Stack Used

TensorFlow

ubuntu

OpenCV

GitHub

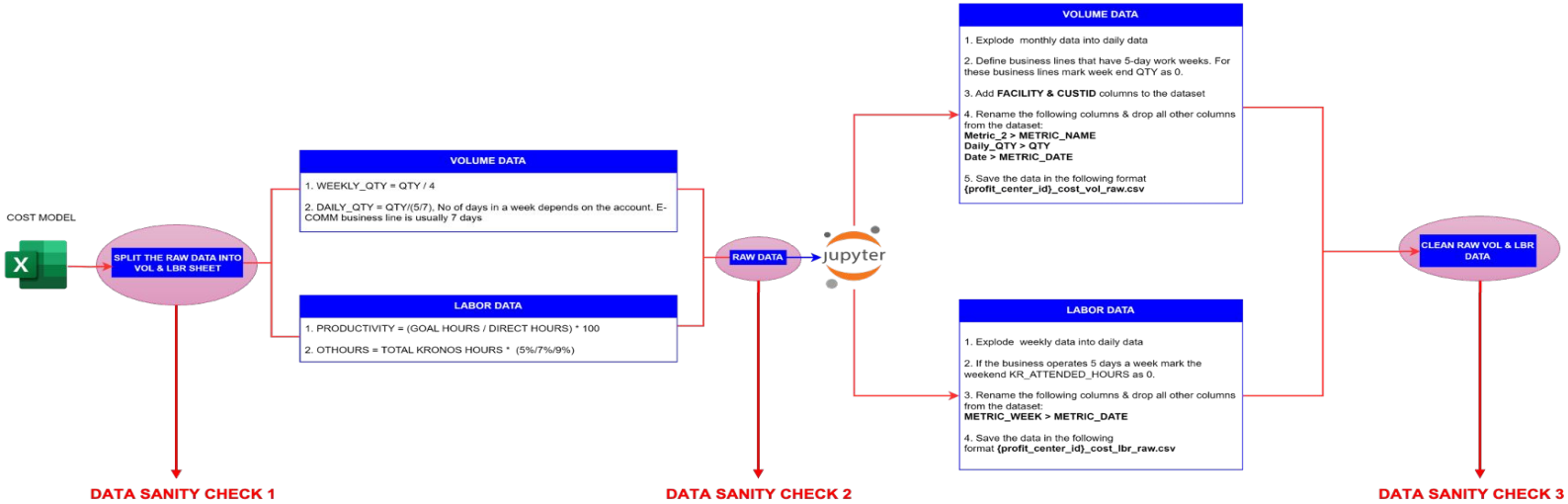
ROS

TURTLEBOT3

Summer 2023 Internship Project

Optimizing Workforce Management at GEODIS with Data Science

- **Background:** Geodis, a global 3PL firm, manages peak season logistics for prominent clients including Apple, Disney, and LEGO.
- **Challenge:** Addressing substantial order volume surges during two annual peak seasons - Black Friday and Christmas.
- **Objective:** Develop a Time Series Forecasting model to predict daily temporary labor requirements during peak seasons across various warehouse locations.
- **Model:** Utilized Facebook's Prophet library for accurate and reliable labor forecasting.
- **Impact:** The tool significantly aids HR and Operations in strategic (yearly & monthly) and tactical (weekly & daily) planning, ensuring optimal labor allocation for each customer and warehouse.



- **Objective:** Streamline integration of new accounts into the forecasting tool.
- **Data Management:** Historically sourced data from Oracle Data Warehouse.
- **Challenge:** New accounts lacked existing database entries.
- **Code Adaptations:**
  - Enabled data loading from .csv for new accounts.
  - Ensured smooth transition to database loading post-account activation.
- **Data Workflow:**
  - Executed rigorous data handling: loading, splitting, cleaning, and sanity checking, as illustrated in the flow diagram.

REQUIREMENTS

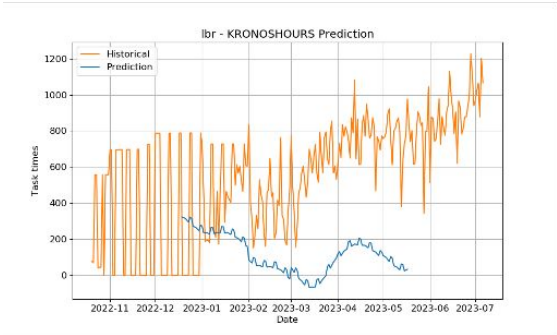
- Utilize data from .csv files for start-up accounts initially.
- Transition to direct data querying from the Oracle database post-account activation.

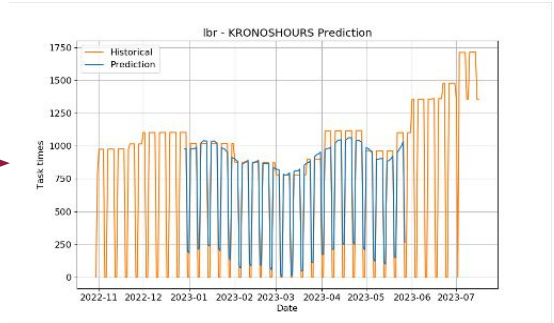
CHALLENGES

- Not all business lines (ecomm, retail etc) for a particular account go live on the same date.
- This creates wild fluctuations in the data, as well as the projections.

SOLUTION

- Engaged with SEMs from all startup accounts to determine provisional stabilization dates.
- Updated "account_start_date.csv" with acquired data.
- Employed cleaned cost model data until stabilization, then transitioned to live data via an SQL server.





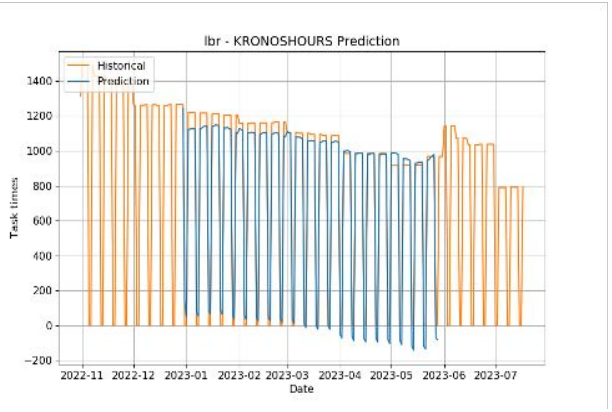
Data loading is completed.

The .csv file is converted into a dictionary.

The forecast tuner performs hyperparameter tuning, automatically fitting the historical data inputs of the model with yearly, weekly, and daily seasonality.

The tuner generates the best set of parameters for each profit_center and different metrics (e.g., SHIPPED, RECEIVED, KRONOSHOURS).

```
{'KRONOS_HOURS':            ds     y   METRIC_NAME
0    2020-02-01   0.0  KRONOS_HOURS
1    2020-02-02   0.0  KRONOS_HOURS
2    2020-02-03  59.0  KRONOS_HOURS
3    2020-02-04  59.0  KRONOS_HOURS
4    2020-02-05  59.0  KRONOS_HOURS
...         ...   ...           ...
1277 2023-08-01  45.0  KRONOS_HOURS
1278 2023-08-02  45.0  KRONOS_HOURS
1279 2023-08-03  45.0  KRONOS_HOURS
1280 2023-08-04  45.0  KRONOS_HOURS
1281 2023-08-05   0.0  KRONOS_HOURS

[1282 rows x 3 columns], 'OTHOURS':            ds    y METRIC_NAME
0    2020-02-01  0.0     OTHOURS
1    2020-02-02  0.0     OTHOURS
2    2020-02-03  9.0     OTHOURS
3    2020-02-04  9.0     OTHOURS
4    2020-02-05  9.0     OTHOURS
...         ...  ...         ...
1277 2023-08-01  4.0     OTHOURS
1278 2023-08-02  4.0     OTHOURS
1279 2023-08-03  4.0     OTHOURS
1280 2023-08-04  4.0     OTHOURS
1281 2023-08-05  0.0     OTHOURS

[1282 rows x 3 columns], 'PRODUCTIVITY':            ds     y   METRIC_NAME
0    2020-02-01   0.0  PRODUCTIVITY
1    2020-02-02   0.0  PRODUCTIVITY
2    2020-02-03  11.0  PRODUCTIVITY
3    2020-02-04  11.0  PRODUCTIVITY
4    2020-02-05  11.0  PRODUCTIVITY
...         ...   ...           ...
1277 2023-08-01  20.0  PRODUCTIVITY
1278 2023-08-02  20.0  PRODUCTIVITY
1279 2023-08-03  20.0  PRODUCTIVITY
1280 2023-08-04  20.0  PRODUCTIVITY
1281 2023-08-05   0.0  PRODUCTIVITY
```

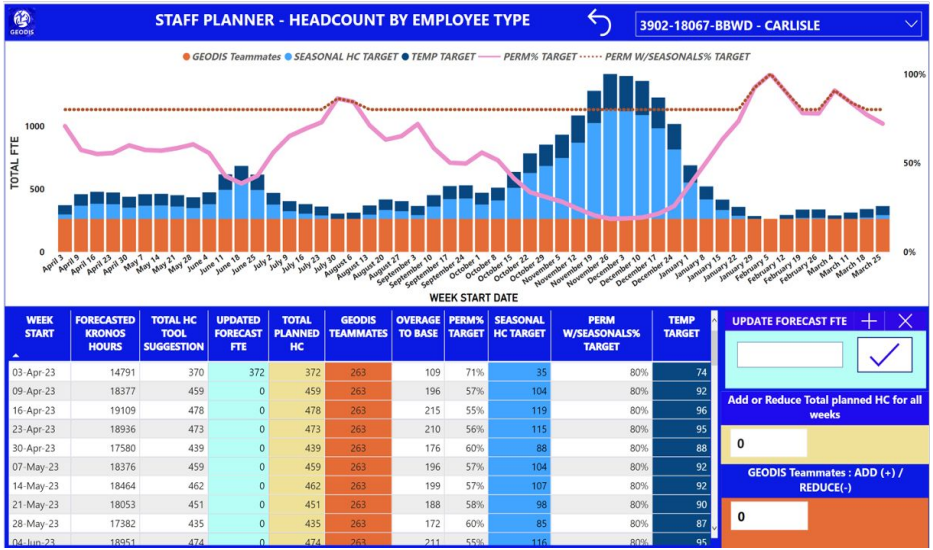



```
"0514_18244_SHIPPED": {
    "features": [
        "quarter_number",
        "is_monday"
    ],
    "conf_interval": 0.9,
    "prior_scale": 1.5,
    "holidays": 1,
    "yearly_seasonality": "auto",
    "quarterly_seasonality": 1,
    "monthly_seasonality": 1,
    "weekly_seasonality": 0,
    "hourly_seasonality": 0,
    "seasonality_mode": "multiplicative",
    "masking_features_train": [],
    "masking_features_pred": []
},
```

Conclusion

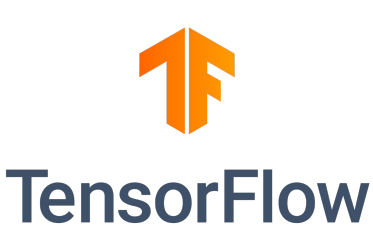
Successful Project Completion: Key Highlights

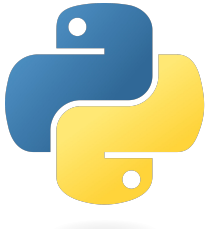
- Conducted in-depth Exploratory Data Analysis (EDA) for major accounts like LEGO and Disney, utilizing advanced feature engineering and seasonality adjustment.
- Executed vital code modifications to ensure data compatibility with new accounts.
- Upheld code reliability through meticulous testing and debugging.
- Developed a tactical time series model, aiding in forecasting for over 100 customers and enhancing daily operations.
- Delivered an interactive PowerBi dashboard, enabling informed decision-making across various teams.



PowerBi dashboard

Tech Stack Used





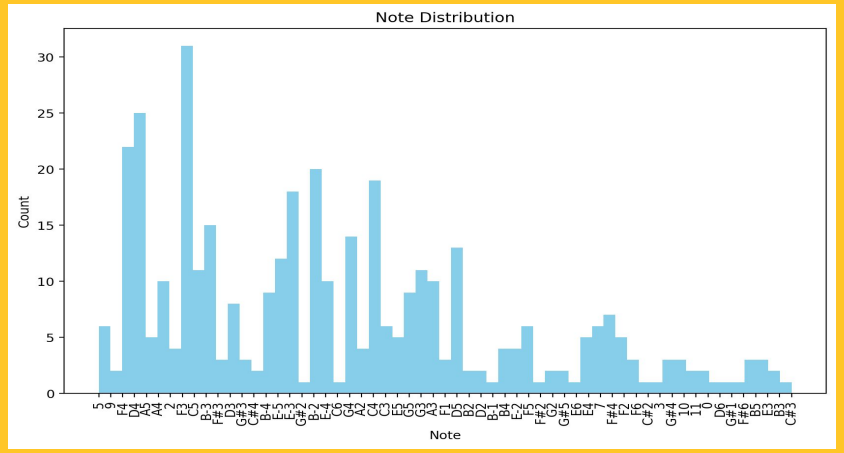
WinSCP

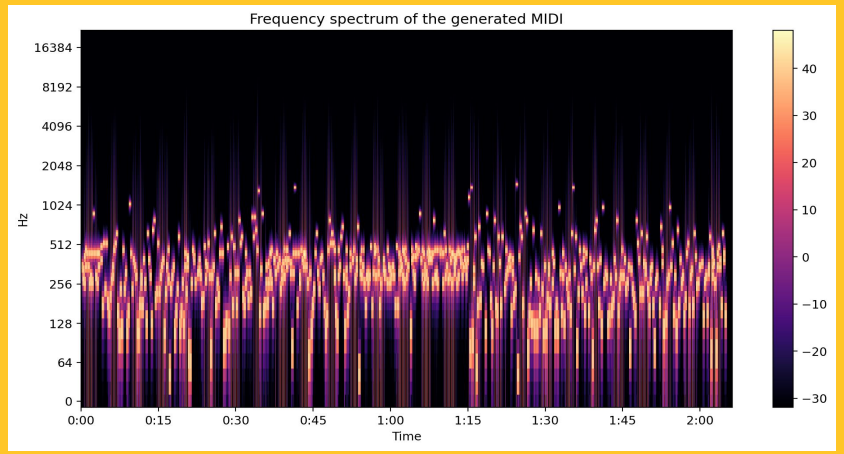
Music Generation using LSTM

Github: https://github.com/iamharkirat/music_generation_LSTM.git

Striking a Chord with LSTM: Automated Melody Crafting Using Neural Networks

- **Objective:** Develop a melody prediction model using LSTM to generate harmonious music based on initial melody input.
- **Dataset:** Utilized 70 Beethoven .midi files to train the model.
- **Encoding:** Transformed .midi files into mapped integers to navigate through the neural network.
- **LSTM Model:** Implemented Long Short-Term Memory networks for predicting subsequent keys and notes in a melody.
- **Melody Prediction:** Successfully predicted keys and notes by merely initializing the melody input.
- **Harmonious Outputs:** Ensured the generated melodies were coherent and harmonious upon testing with various melody inputs.





Thank You!