# Traffic Object/Lane Detection

*Critical Design Review*

2A1 (OLD482)

Khai Nguyen
Viet Nguyen
Pavan Poladi
Jason Bernal

Department of Computer Science
Texas A&M University

10/30/2022

# Table of Contents


1 Introduction 3

2 Proposed design 4

2.1 Updates to the proposal design 4

2.2 System description 7

2.3 Complete module-wise specifications 9

3 Project management 14

3.1 Updated implementation schedule 15

3.2 Updated validation and testing procedures 18

3.3 Updated division of labor and responsibilities 19

4 Preliminary results 21

# 1 Introduction

## 1.1 General Scope, Problem Background, and Needs Statement

This ROS framework currently contains a separate lane detection framework and a separate object detection framework. This creates a decrease in efficiency, usability, and simplicity as one would need to run their models through both detection frameworks separately. By having these be two separate detection models, this also leaves more room for discrepancies between the two models as well as preventing future growth from not being able to combine shared results and data from a joint detection model to get unique perspectives. We as a team need to make ROS' use of object and lane detection more efficient, accurate, and intuitive. Ideally, we need to essentially replace the current system which has a separate lane detection framework and object detection framework, so that ROS can reek all the benefits of having a combined lane and object detection framework, including being able to make future improvements more easily. It is important to note that there are a few models we can use which have joint detection frameworks including Yolov6 and HybridNets for instance. Our plan is to to integrate HybridNets into the pipeline, and perhaps try to integrate another model if time permits or if HybridNets doesn't integrate well enough to meet our expectations.

## 1.2 Goal and objectives

Our goal is to create a clear and flexible pipeline allowing ROS to integrate an existing combined lane and object detection framework so users of ROS that are interested in this field can have a more smooth, simple, and confident experience when implementing their work. Ideally, we want to create a system that is more efficient, more accurate, and more flexible than the current ROS pipeline. Our first objective is to first fully understand the ROS framework so that we can truly understand how to take advantage of key features of ROS to help reach our goal. Our second objective is to begin testing ROS' current lane detection and object detection systems to see how efficient they are, how accurate they are, how intuitive and flexible they are, and how easy it is for a new user to begin integrating these systems. Our third objective is to try out the existing combined lane and object detection framework that ROS is not currently using to see how efficient, accurate, and intuitive it is compared to ROS' separate lane and object detection systems. This step will also allow us to evaluate the pros and cons of integrating this system as well as how compatible it is with ROS' framework. Our final objective is to develop a pipeline using the ROS framework and the existing combined lane and object detection framework to allow a user to take an image as an input and return an image with bounding boxes and classification results, all within the ROS framework. It is important to note that we are determined to achieve out objectives with zero additional spending and only using our own physical computing machines/laptops to run the complex algorithms and large data sets to emulate how a typical new ROS user would eventually use the integrated pipeline.

## 1.3 Design Constraints and Feasibility

There are many constraints we have to consider when evaluating the feasibility of our project objectives. One technical constraint is that our team does not have great experience in the topic of machine learning and graph architectures beyond the knowledge we have gained from a basic data structures and algorithms course. This is necessary to truly understand the ROS framework as well as the existing lane and object detection framework. The more familiar we are with these concepts, the more likely we will be able to make a more efficient and intuitive pipeline that will take full advantage of ROS' capabilities. Another technical constraint is the lack of knowledge on robotics within our team. Having a good knowledge of robotics would allow our team to consider other possibilities for our pipeline and have a better understanding of how it can actually be used in various scenarios involving robotics beyond just autonomous and non-autonomous vehicles. Another technical constraint is that ROS has been being

developed for years and we simply have never used the framework. We would be in a much better position and have much higher chances of meeting our objectives if we had prior experience using ROS or similar frameworks. This leads us to a major temporal constraint. We only have about 10 weeks starting from the day of our project proposal presentation to meet our objectives. With more time, we would be able to create a much better pipeline and system which would be far more likely to actually be integrated into the ROS framework in the future. We also have a major physical constraint being that we are only a team of 4 computer scientist undergraduates. Logically speaking, if we had a larger team or members that were more experienced, we would simply be able to achieve much better results and have higher chances of meeting our objectives. Another physical constraint is the performance capabilities of our computers. The performance and amount of testing we are able to achieve is highly determined on how capable our systems are to run these complex algorithms on large data sets. Lastly, we have to consider economic constraints. Many of the existing lane and object detection frameworks out there are free for the public to use, but perhaps there are paid frameworks which could be far better for the purpose of reaching our goals. We want to achieve objectives with zero additional spending, which could limit the possibilities of the overall framework achieving the greatest success.

### 1.4 Validation and Testing Procedures

It is important to remember that this project would be a waste unless we do our due diligence and validate that we have improved upon the current system and didn't just reinvent the wheel. In order to do this we will have to run a variety of tests. The first deliverable we must validate is making sure we have made robust and valuable comparisons between the current ROS pipeline that uses separate lane detection and object detection models and the external framework we were provided with that uses a combined lane and object detection model. We will make our own comparisons and validate these results by doing extensive testing using data samples on each framework. This will allow us to back-up our findings and validate our comparisons on the two frameworks on both ideal and non-ideal conditions. The second main deliverable we must validate is that the new pipeline we built actually works as intended. This pipeline will use ROS nodes to take in images as an input, call the functions present in the external framework that uses a joint lane and object detection model, and return the new images with bounding boxes and classification results. We will be testing each of these sections by comparing the inputs and outputs from ROS and the joint detection framework at every step of the way.

## 2 Proposed design

### 2.1 Updates to the proposal design

With HybridNets being a new machine learning framework which was released on March 13, 2022, the developers continually add new features on a rolling basis.

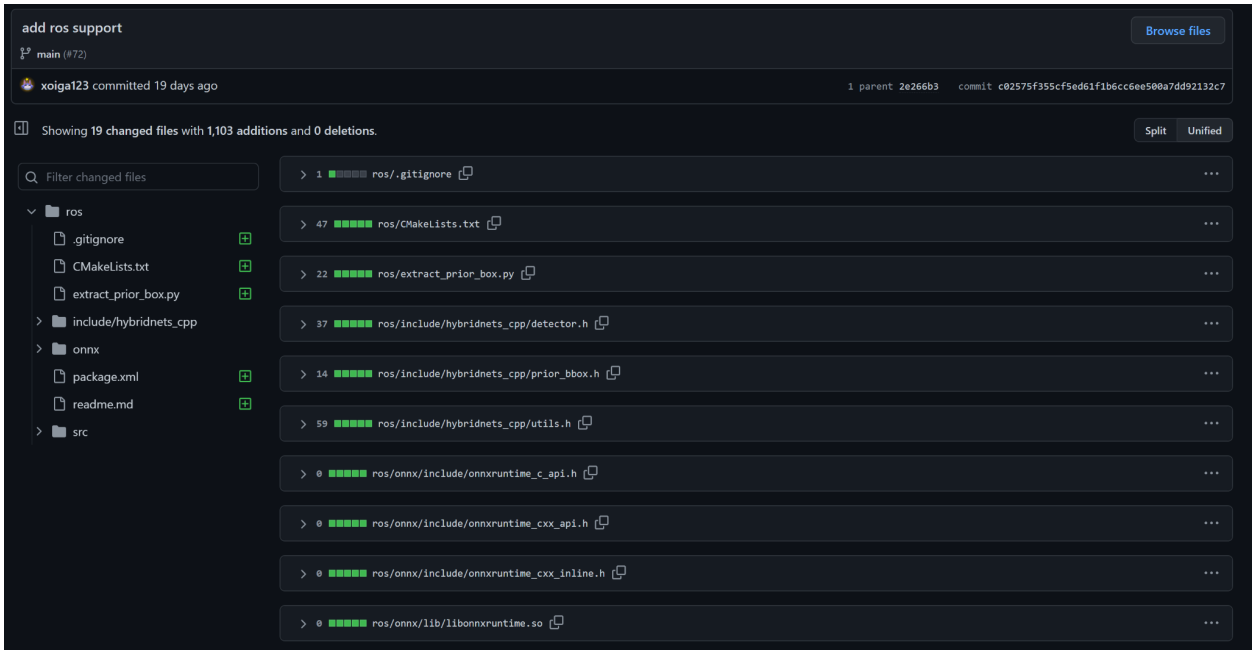



*Figure 1: New ROS Support Added*

As seen on Figure 1, the team discovered that the developers had released some level of ROS support on Oct. 11th, 2022. The contribution made is considered an Open Source Contribution. The support for ROS was made through the use of Open Neural Network Exchange (ONNX), a machine learning library. The team consulted the Professor and TA to ensure that the use of the discovery is allowed. Since the feature is Open Source, the team was allowed to explore the new finding. There was proper documentation on having the feature up and running on a new machine but the team found that the developers had created an exclaimer explaining that the ROS support is hard-coded and may need to be fine tuned on a case-by-case basis. The team went through what was added to determine the level of complexity that is required in order to replicate similar results. The team attempted to run the new ROS support on a local machine but with too many dependencies and installations, the team were unable to fully run the feature. The team concluded that the added ROS support is too experimental and has not been fully tested so the team continued to develop their own implementation of ROS support. Although the new discovery did not pan out as the team had expected, the team was able to gain a better understanding on how to tackle the problem at hand. The team had abstracted some of the architecture behind the feature to aid in the development of the team's own solution.

| **Item** | **Cost** |
|---|---|
| External Hard drive | $50 - $100 depending on how big of files we need and such like datasets |
| Google Colab Pro Subscription | $9.99 / month. 100 compute units per month |
| Labor Hours | 10hr/Week each for 10 Weeks: 400hrs |

*Table 1: Updated Team Budget*

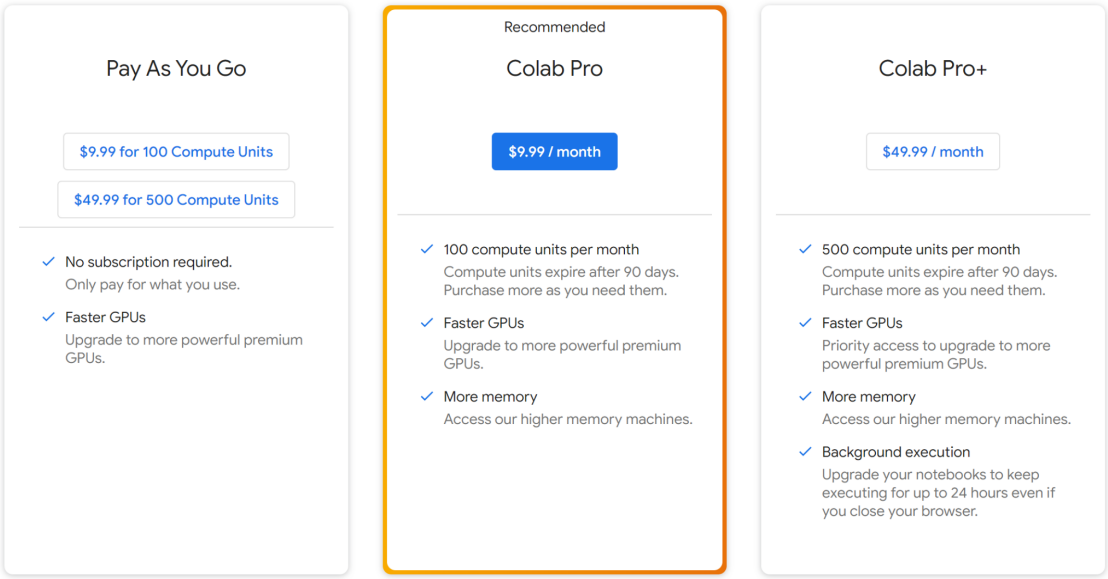



*Figure 2: Google Colab Subscriptions*

As seen in Table 1, a change to the budget was necessary to aid in the development process of the team. The Google Colab software has been able to aid in the development of the project due to the computational resources it offers compared to the local machines that the team currently own. Figure 2 shows the various Google Colab subscriptions that are offered. The team has purchased the Google Collab Pro Subscription which is a monthly expense. With the team only having a timeline that fits within the semester, typically four months, the team found the price to be cost-effective. The subscription allows for additional compute units, faster GPUs, and more memory.

The team also has accounted for the labor hours that have gone into the project thus far. The team estimates that the entire team averages about 40 hours of work per week, 10 hours individually each, which would equate to a total of 400 hours to fully complete the project. The labor hours required to complete such a project with Software Engineers is the biggest expense the project has encountered. It is hard to estimate the value regarding the number of hours that has been spent so far in the project but as suggested, it is the biggest expense the project has seen.

We do not foresee this project costing too much in terms of money because all datasets are given for free, the fact that the project is mostly software, and that APIs are free of cost. The only potential cost that we might see is that we might choose to purchase an external hard drive to use the powerful computer we have access to in our lab room. This computer is much more computationally powerful than any of our team member's laptops, which means it will allow us to run all of the necessary models and tests we create much faster. We will outweigh the pros and cons of using this system after running some of the models and tests on our own computers first.

Perhaps the most important thing in wasted costs is that since HybridNets is during ongoing development, we may have to be concerned with how new updates could change the performance or dependencies of our project.

## 2.2 System description

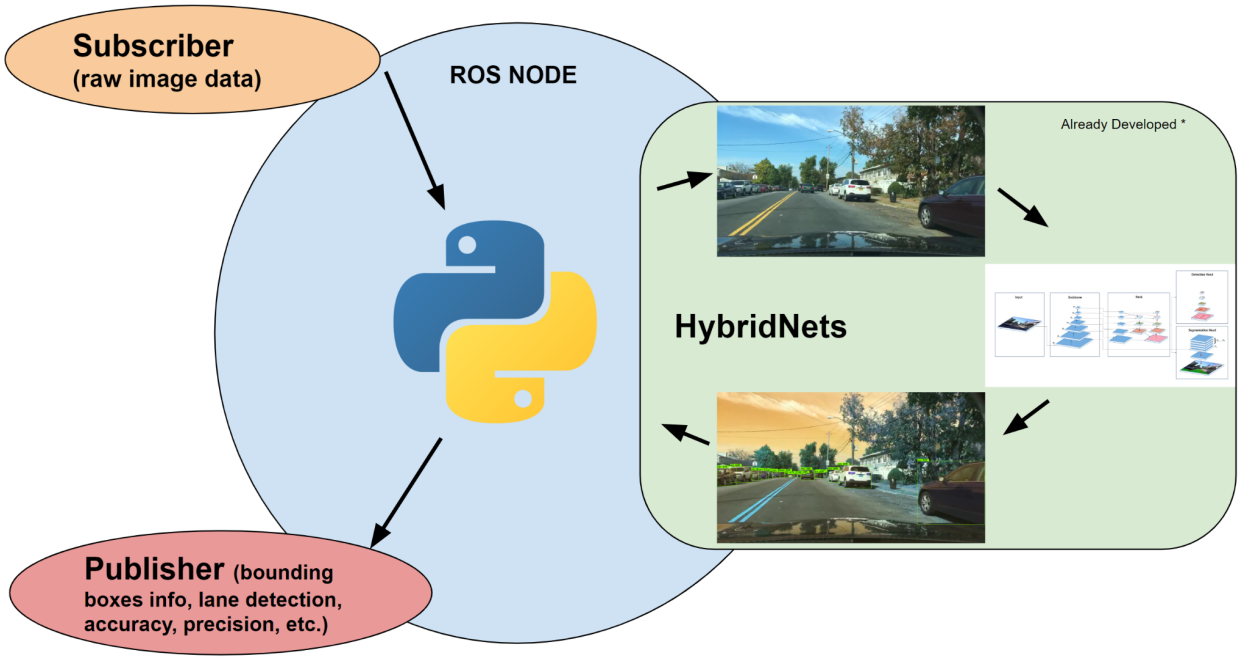



*Figure 3: High Level Block Diagram*

Figure 3 shows an in-depth look at what the team is aiming to accomplish. The diagram shows the node in which the team's python script will carry out a variety of tasks. The node first will receive raw image data due to subscribing to a node which outputs image data. It's important to note how the HybridNets model works. The HybridNets model takes an image and then executes its machine learning algorithm as as shown in the figure, then it outputs the image along with other metrics such as bounding boxes and accuracy above the objects. The figure above shows an example of a Python script but this is the main objective that the team is developing on. Not only should the team ensure that their Python script is able to route the unprocessed imaged to become an image which has the object/lane detection, the team should also work in a Docker environment to ensure that the Python script can work on any machine. The blue node in the figure can also represent the Docker environment in which the Python script that the team develops will work in. Docker is very useful due to the fact that the developed scripts can be packaged and used across multiple operating systems, eliminating case where it may work on one computer and not another. The team has been able to make significant progress on Python script and is working on ensuring that there developed software is defect free. As shown in the figure, the green block represents the HybridNets model, which has already been developed by other developers. The team's script must be able to communicate well with ROS and the HybridNets machine learning model which helps detect Objects/Lanes. One critical consideration that the team is actively working on is the compatibility between the images that are received from the subscriber and ensuring that the correct image format is made when publishing the information to additional ROS nodes.

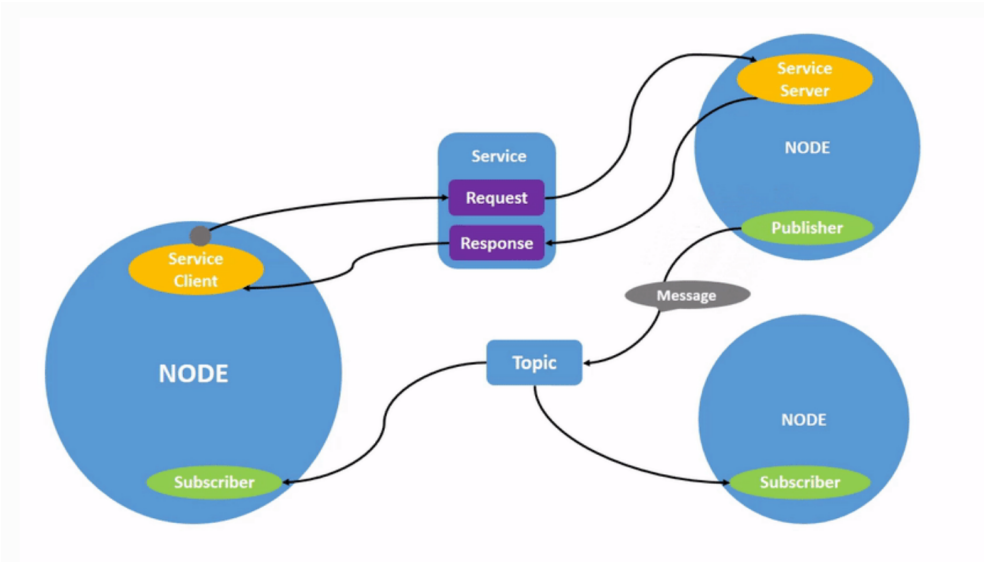



*Figure 4: ROS Architecture*

As shown in Figure 4, the architecture behind ROS is a graph-like structure where nodes are processes and edges illustrate the flow of communication between nodes. In order for the team to complete the project, the team will have to create a ROS model that will better meet the needs of the project. The team formulated Figure 5 (*High Level ROS Architecture)* to better fit the scope of this project.

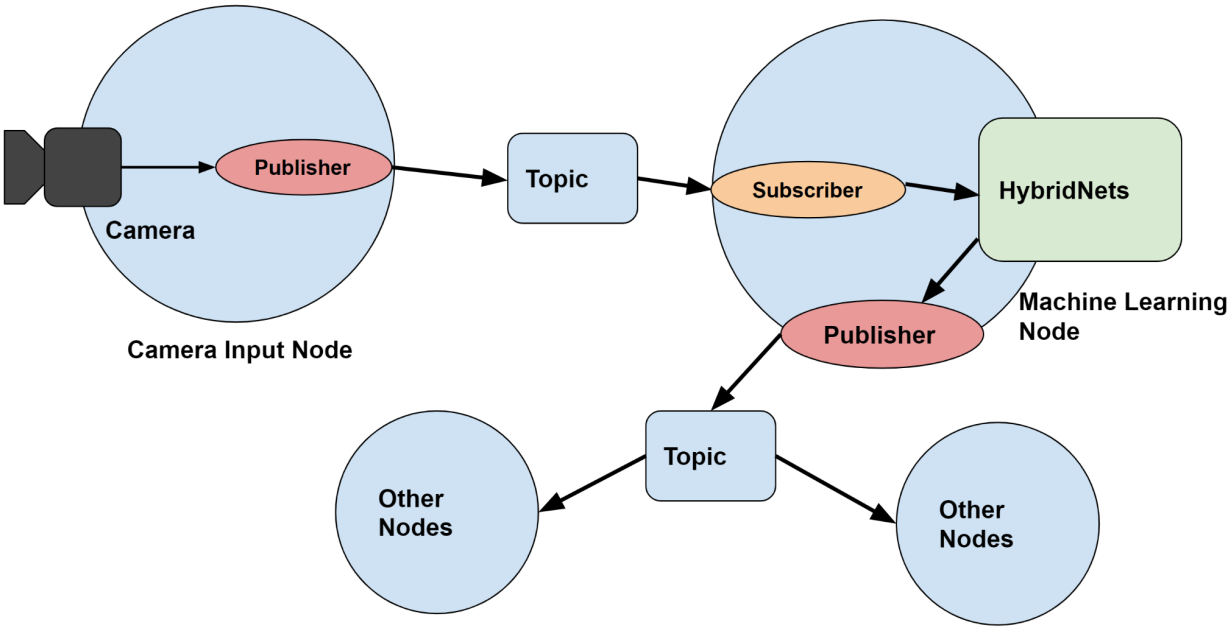



*Figure 5: High Level ROS Architecture*

Utilizing ROS Architecture, the team will have the machine learning model be incorporated into a ROS node. The integration of the model to the node is the main challenge of the project as the team has limited experience of the technologies at hand. The team will look into how previous models have been integrated into ROS and see if there are any libraries or tools that may aid in the integration process. Team members Jason and Khai do have experience working with Machine Learning models but ROS is a new software

for all members of the team. Jason and Khai will use the foundational knowledge they learned in the classroom to better help the team in the machine learning aspect of the project. As discussed earlier in this document, the machine learning model the team will plan on integrating will be the HybridNets model due to its high accuracy and performance. The machine learning model will be placed in one node of the ROS model. Each node is meant to include only one process/purpose and thus the model being in one node is the appropriate design choice. The team will have an additional node which will capture real time information of a vehicle driving. The camera will capture video of a vehicle driving and will be the raw data that the team will use for the project. The datasets will be provided to the team so the team does not need to capture their own datasets. Figure 5 demonstrates the ROS node that is receiving input from the camera will communicate that information to the node with the Lane/Object Detection framework. The Lane/Object Framework will then utilize the Machine Learning framework to detect lanes and traffic objects. The results of the detection will then be sent to other nodes through the topic IPC method. A response will also be sent to the camera node to validate that the request was successfully processed. A topic method is used due to the fact that it is mainly used to process continuous data streams. The service method is used to be able to send a response from the node that is sending information. The other nodes are beyond the scope of this project, the main purpose of the project is to ensure that the raw input data is processed correctly. As mentioned, the exact machine learning model the team will implement will be the HybridNets model but the team will consider other models as each has their own advantages and disadvantages. A stretch goal for the team is to be able to incorporate various machine learning frameworks and do comparative testing to see which one would be best overall. However, to meet the acceptance criteria of the project, the team will integrate the HybridNets model at minimum. Regarding additional information about the nodes in ROS, it is important to note that each node has a logger. The team will utilize the logger to aid in identifying any bugs or defects that may arise from the integration process of the framework. Once the team has completed the initial integration of the model, they will then begin to do performance and comparison testing on different types of machine learning models to see which one performs best overall.

### 2.3 Complete module-wise specifications

Our team decided to optimize the current recommendation framework named HybridNets: End2End Perception Network developed by Dat Vu, Bao Ngo, and Hung Phan[10].. Computer vision for auto driving is not new in the field. Besides HybridNets, we have many other state-of-the-art networks such as: YOLO, UNet, SegNet, ERNet, LaneNet, and SCNN for lane detection and segmentation. We chose this framework, because the model claimed that they improved the speed and performance thanks to its architecture. The core concept of this architecture is to create an encoder-decoder architecture, where the backbone and neck generate a context for producing end-to-end visual perception.

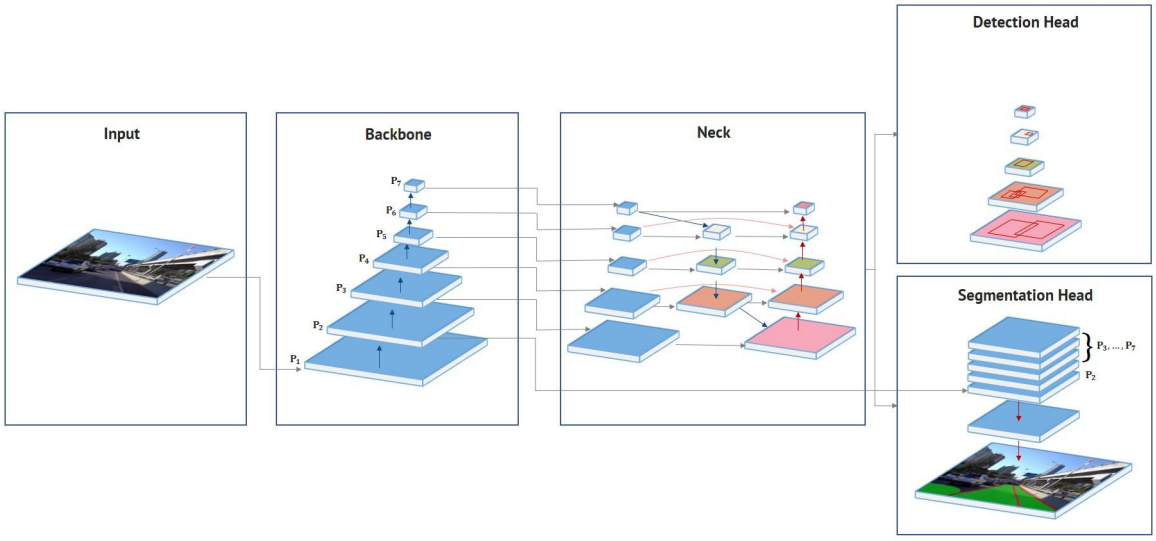



*Figure 6: Machine Learning Model*

**General Design:**

In auto driving, multi-tasking is very important. Autonomous driving must have the capacity to detect humans, weather conditions, lanes, obligations, traffic signs….Therefore, our team needs a network that can do the end-to-end multi-task visual perception. This is the core concept of HybridNets. According to the paper, the HybridNets network does traffic object detection, drivable area segmentation, and lane detection segmentation.

To achieve these tasks, the network consists of an encoder and two different decoders for multitasking. In the encoder, there are the backbone and neck. In the decoder, one is for segmentation, and the other is for detection.

**Encoders:**

For the whole network, the speed of the encoder is crucial because the detections and segmentation all relied on the features of the encoder.

For the backbone, we can see from Fig. 6 that the P3 to P7 are reduced consistently in size with a resolution of $1/(2^i)$ of the input image. For example, if the input resolution is 896x384. In P2, the resolution will become $(896/2^2, 384/2^2) = (224,96)$, while in P7 it would become (7,3). They implemented these by using EfficentNet neural network models as the model proved to be efficient by its optimization in depth, width, and resolution parameters with lower computation cost. These features in the backbone will go to the neck network pipeline after extraction.

For the neck, the most challenging part is the multi-scale feature that is received from the backbone. In this paper, they use BiFPN based on EfficientDet for better performance, because it is popular in fusing features at different resolutions top-down and bottom-up and add weight to each feature to learn the importance of each level.

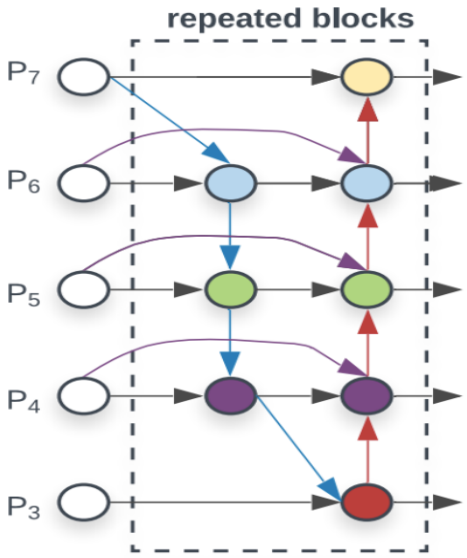



*Figure 7: The BiFPN block used in neural networks.*

**Decoder:**

As mentioned, the detection and segmentation heads are all in the decoder of the hybridNets neural network.

In this neural network, they use the idea of anchor boxes for detection. Each feature map from the neck has assigned nine prior anchor boxes with different aspect ratios. Similar to YOLOv4, they use K-means Achor clustering with nine clusters with three different scales for each grid scale in addition. In the end, the detection phase will predict the offsets of the bounding boxes from small to large. With this design, the network can smoothly on the complex dataset, and then, in the final steps, it assigns the probability of each class, and also the confidence of the prediction boxes.

In the segmentation head, they would produce three classes: background, drivable area, and lane lines. . Then, the P2 features will go through ConvBlock and meet all other features which already go through BiFPn. All Ps will be combined in the final steps to restore the final features with the original size of the image. This is where this network stands out as it helps the network improve output precision.

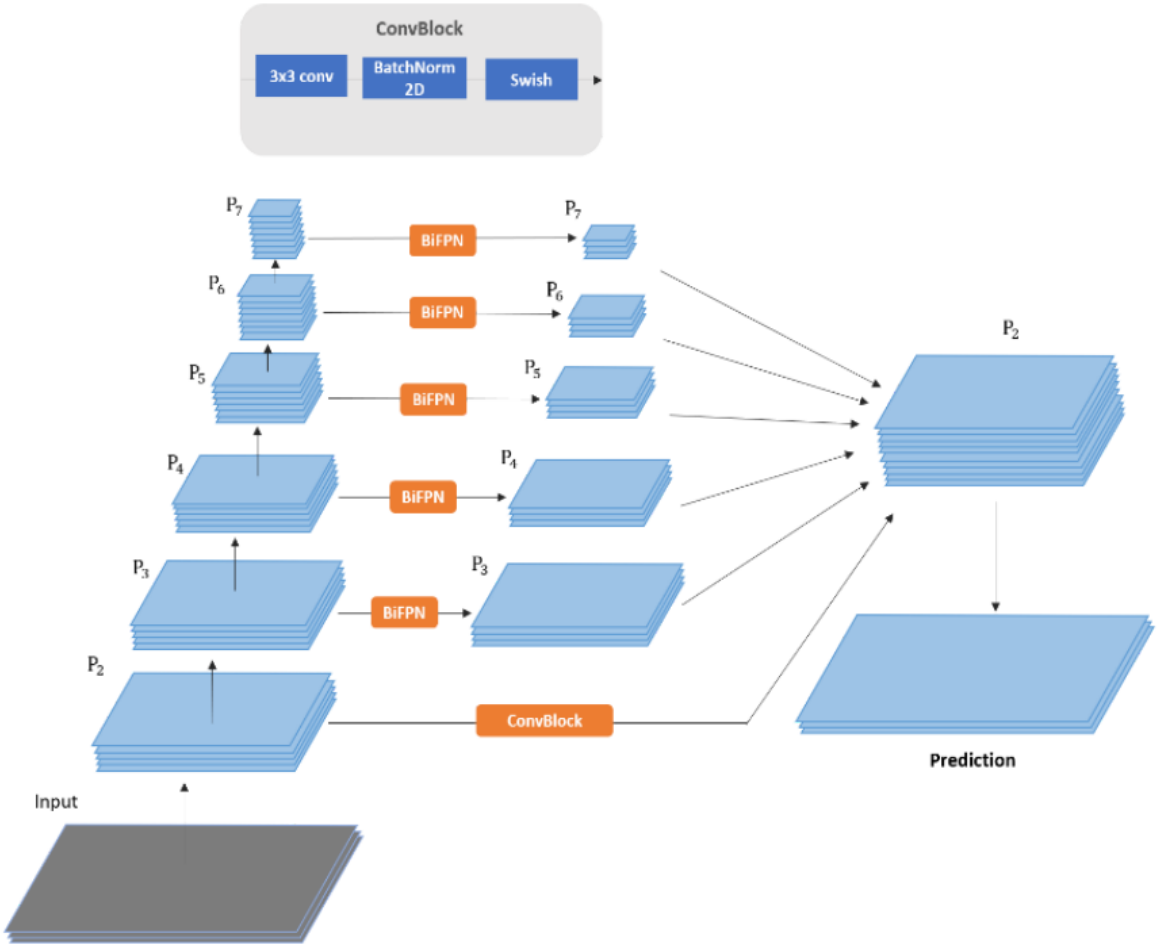



*Figure 8: Neck architecture*

Algorithm: The author resizes all images with resolution 640x384 before training them in the network. This resolution is a trade-off between performance and accuracy, and the dimensions are also divisible by 128 for the BiFPN. This algorithm is the logic that makes the system work.

**Algorithm 1**. HybridNets training stage. First, we only train Encoder and Detection Head as object detection task. Second, we freeze the Encoder, Detection head and unfreeze parameters from Segmentation Head. Finally, the final network is trained jointly for all tasks.

**Input:** Target end-to-end network $\mathsf{F}$ with parameter group $\Theta = \{\theta_{enc}, \theta_{det}, \theta_{seg}\}$;

Training dataset $\mathsf{T}$;

Threshold for convergence $\gamma = \{\gamma_1, \gamma_2, \gamma_3\}$;

Total loss function $\mathsf{L}$;

Pivot strategy $\mathsf{P} = \{\{\theta_{enc}, \theta_{det}\}, \{\theta_{seg}\}, \{\theta_{enc}, \theta_{det}, \theta_{seg}\}\}$

**Output:** Proposed network: $\mathsf{F}(\mathsf{X}, \Theta)$

**procedure** Train $(\mathsf{F}, \mathsf{T})$
  **for** $i = 0$ to length $(\mathsf{P})$ - 1
    $\Theta \leftarrow \Theta \cap \mathsf{P}[i]$ // Freeze parameters
    **repeat**
      Sample a mini-batch $(\mathbf{x}_m, \mathbf{y}_m)$ from training dataset $\mathsf{T}$
      $\ell \leftarrow \mathsf{L}_{all}(\mathsf{F}(\mathbf{x}_m; \Theta), \mathbf{y}_m)$
      $\Theta \leftarrow \arg\min_{\theta} \ell$
    **until** $\ell < \gamma[i]$
    **if** $i <$ length $(\mathsf{P})$ - 1 **then**
      $\Theta \leftarrow \Theta \cup \mathsf{P}[i+1]$
    **endif**
  **end for**
**end procedure**
Train $(\mathsf{F}, \mathsf{T})$
**return** Proposed network $\mathsf{F}(\mathsf{X}, \Theta)$

*Figure 9: HybridNets Algorithm*

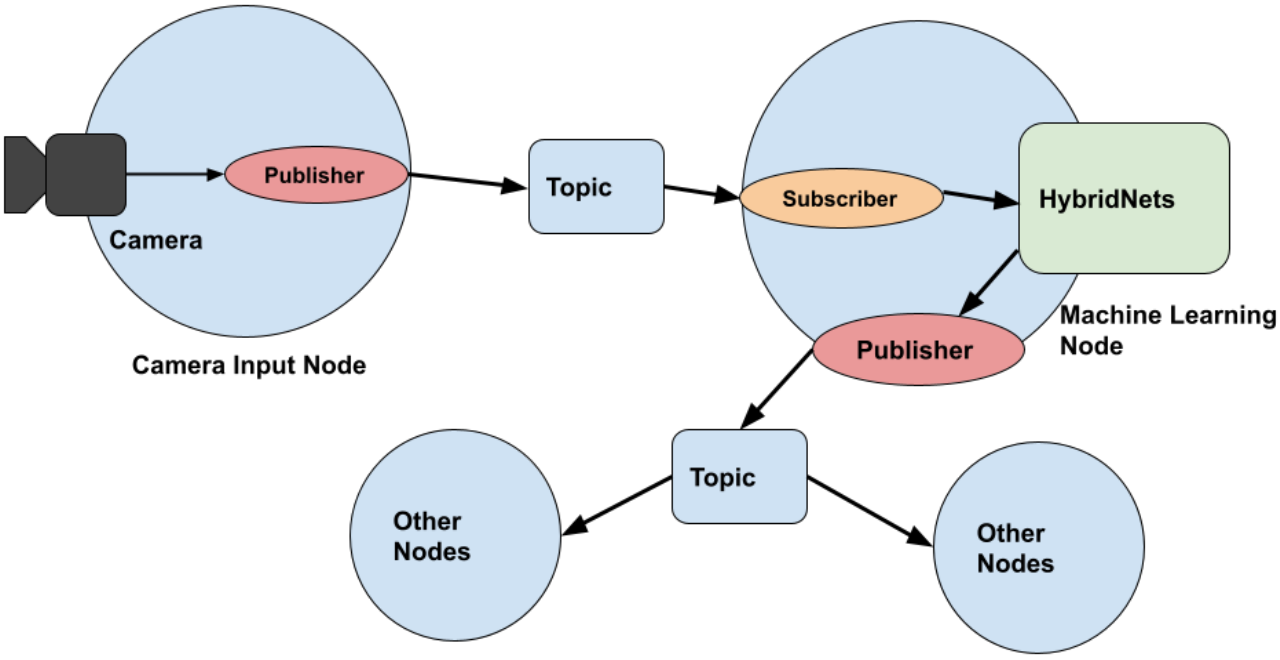



*Figure 10: High-Level ROS Architecture*

The architecture behind ROS is a graph-like structure where nodes are processes and edges illustrate the flow of communication between nodes. ROS nodes send requests for data and also receive data via responses. This is achieved through ROS' clever and sophisticated use of subscribers, publishers, and topics. Subscribers are nodes that are written to obtain the messages and information being published on a ROS topic. Publishers send the messages in a variety of message types to specific ROS topics. Topics define the types of messages that will be sent. In order for the team to complete the project, the team will have to create a ROS model that will better meet the needs of the project. The team formulated the below structure to better fit the scope of this project.

## 3 Project management

Jason Bernal has worked at Charles Schwab as a Software Engineer Intern during Summer 2022. Jason aims to utilize the skills and collaboration he has learned through his internship and apply it to his team. Jason was selected to be the team leader for this capstone project. Jason and Khai have completed the Machine Learning course at Texas A&M University and thus are familiar with Machine Learning models. As a team lead, Jason aims to assist in all parts of the project.

Khai Nguyen has worked at Lynntech Inc. working with various technologies most notably, Computer Vision. As mentioned, Khai has an academic background in Machine Learning and will apply that knowledge to this project. Khai's credentials and experience has given him a competitive advantage of being the most familiar with the current technologies the team is tasked to work with. With such a background, Khai will be taking on the responsibility of system design. From high-level to low-level design, Khai will look into what is feasible given the time constraints imposed on the team. Khai has begun making suggestions as to how the overall architecture of the project will go.

Pavan Poladi has worked at Microsoft as a Program Management Intern. Through Pavan's work experience, he has been able to develop exceptional communication skills. Pavan is currently taking the Machine Learning class offered at Texas A&M University and will be able to apply the knowledge he gains in the classroom over the course of the semester. Pavan is assigned the responsibility of software design. Pavan enjoys solving complex problems and being incharge of software design will be a great opportunity for him to master software engineering.

Viet Nguyen has such a strong academic background in Computer Science that he currently serves as a Peer Teacher at Texas A&M University. Viet has strong communication skills as well and is able to quickly think on his feet. Viet is assigned the responsibility of testing and validation. Viet loves analyzing systems and thus this is a great opportunity for him to work with a large-scale application. Viet will also work in other aspects of the project so that he can continue to grow as a software engineer.

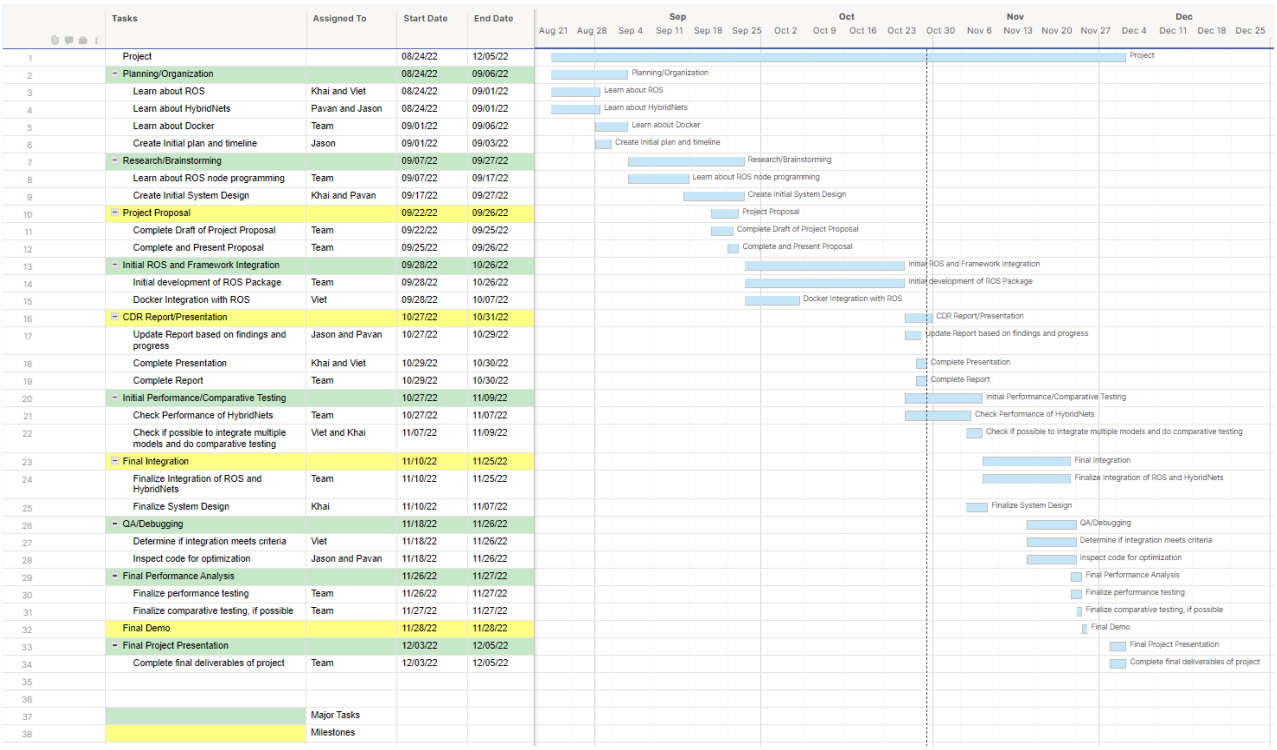
Each member of the team is responsible for technical reporting. The technical reporting will be divided evenly among the team members so that each team member will get exposure to the level of reporting that a project requires. The weekly reports are rotated among the team members every week. Although the weekly report is assigned to a particular individual on the team, each member is encouraged to share the documents and discuss with the team before submitting. The team will revise and give each other feedback regarding our technical reporting to ensure we meet the acceptance criteria required.

Weekly meetings with the Instructor and TA have been vital to the success of this project. The Instructor will oversee the progress of the team and will assist if the team encounters a roadblock. In order to keep track of project progress, the team will create a list of tasks to complete and list them in the weekly report. The weekly report will specify the tasks that need to be completed and the deadline set by the team. Gantt charts will also be utilized in order to visualize the progress of the team. The revised Gantt chart will continue to give a high level overview of the stages of progress the team will encounter over the course of the semester.

The team has not needed to use any additional mechanisms to manage the team or project. The only change that has occurred is that the team has adopted pair programming sessions instead of mob coding sessions in order to make progress on the project. The management approach has been working well for the team. Although progress has been a little slow in terms of the initial Gantt chart that was created, the team is making continual and steady progress. There is great team satisfaction and everyone is able to work well together. The team anticipates that we will continue to work with a positive attitude to meet the deadlines that have been set for this project.

### 3.1 Updated implementation schedule

In order for the team to be able to complete the project within the allocated time, the project will be divided into different stages. The creation of a Gantt Chart will be utilized in order to better represent these stages. The Gantt chart below, Figure 11, will show the stages for this project:



| | Tasks | Assigned To | Start Date | End Date |
|---|---|---|---|---|
| 1 | Project | | 08/24/22 | 12/05/22 |
| 2 | Planning/Organization | | 08/24/22 | 09/06/22 |
| 3 | Learn about ROS | Khai and Viet | 08/24/22 | 09/01/22 |
| 4 | Learn about HybridNets | Pavan and Jason | 08/24/22 | 09/01/22 |
| 5 | Learn about Docker | Team | 09/01/22 | 09/06/22 |
| 6 | Create Initial plan and timeline | Jason | 09/01/22 | 09/03/22 |
| 7 | Research/Brainstorming | | 09/07/22 | 09/27/22 |
| 8 | Learn about ROS node programming | Team | 09/07/22 | 09/17/22 |
| 9 | Create Initial System Design | Khai and Pavan | 09/17/22 | 09/27/22 |
| 10 | Project Proposal | | 09/22/22 | 09/26/22 |
| 11 | Complete Draft of Project Proposal | Team | 09/22/22 | 09/25/22 |
| 12 | Complete and Present Proposal | Team | 09/25/22 | 09/26/22 |
| 13 | Initial ROS and Framework Integration | | 09/28/22 | 10/26/22 |
| 14 | Initial development of ROS Package | Team | 09/28/22 | 10/26/22 |
| 15 | Docker Integration with ROS | Viet | 09/28/22 | 10/07/22 |
| 16 | CDR Report/Presentation | | 10/27/22 | 10/31/22 |
| 17 | Update Report based on findings and progress | Jason and Pavan | 10/27/22 | 10/29/22 |
| 18 | Complete Presentation | Khai and Viet | 10/29/22 | 10/30/22 |
| 19 | Complete Report | Team | 10/29/22 | 10/30/22 |
| 20 | Initial Performance/Comparative Testing | | 10/27/22 | 11/09/22 |
| 21 | Check Performance of HybridNets | Team | 10/27/22 | 11/07/22 |
| 22 | Check if possible to integrate multiple models and do comparative testing | Viet and Khai | 11/07/22 | 11/09/22 |
| 23 | Final Integration | | 11/10/22 | 11/25/22 |
| 24 | Finalize Integration of ROS and HybridNets | Team | 11/10/22 | 11/25/22 |
| 25 | Finalize System Design | Khai | 11/10/22 | 11/07/22 |
| 26 | QA/Debugging | | 11/18/22 | 11/26/22 |
| 27 | Determine if integration meets criteria | Viet | 11/18/22 | 11/26/22 |
| 28 | Inspect code for optimization | Jason and Pavan | 11/18/22 | 11/26/22 |
| 29 | Final Performance Analysis | | 11/26/22 | 11/27/22 |
| 30 | Finalize performance testing | Team | 11/26/22 | 11/27/22 |
| 31 | Finalize comparative testing, if possible | Team | 11/27/22 | 11/27/22 |
| 32 | Final Demo | | 11/28/22 | 11/28/22 |
| 33 | Final Project Presentation | | 12/03/22 | 12/05/22 |
| 34 | Complete final deliverables of project | Team | 12/03/22 | 12/05/22 |
| 35 | | | | |
| 36 | | | | |
| 37 | | Major Tasks | | |
| 38 | | Milestones | | |



*Figure 11: Updated Gantt Chart of Tasks*

The beginning stages of the project involve a considerable amount of planning/organizing and researching/brainstorming ideas on how to tackle the project. ROS is a new type of software for each member of the team and thus will take time for the team to become familiar with. After the team feels more acquainted with the ROS software and Machine Learning framework, the team will move onto initial integration of the systems. It is expected that initial integration will be complex but the team will work together to meet the deadlines. When analyzing dependencies based on the Gantt chart, we can see that Initial ROS and Framework Integration must be completed first before Performance/Comparative Testing can be done. The team will need to find a way to integrate the software and then be able to measure the performance of the machine learning framework. Although comparative testing is a stretch goal for the team, the team will ensure that performance testing is done on the project. The team will report aspects such as accuracy and performance speed of the machine learning framework. Another dependency to consider is the Final Integration and QA/Debugging stages. Final Integration must be complete before the team can move onto QA/Debugging. Once the team has decided on the way the integration will be implemented, the team can move onto testing the integration and ensuring that there are minimal defects. Once the team is satisfied with the correctness of the integration, the team will wrap up the project and present it to the Instructor. The final stages are reserved for completing the deliverables of the project. The Gantt Chart was an essential tool in determining the dependencies of the project.

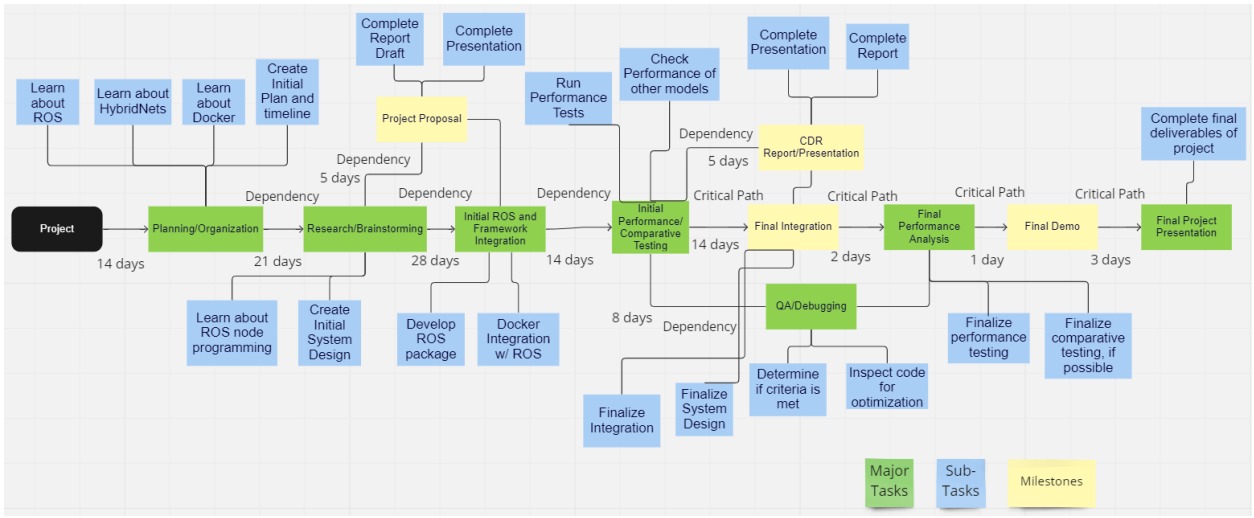



*Figure 12: Full PERT Chart*

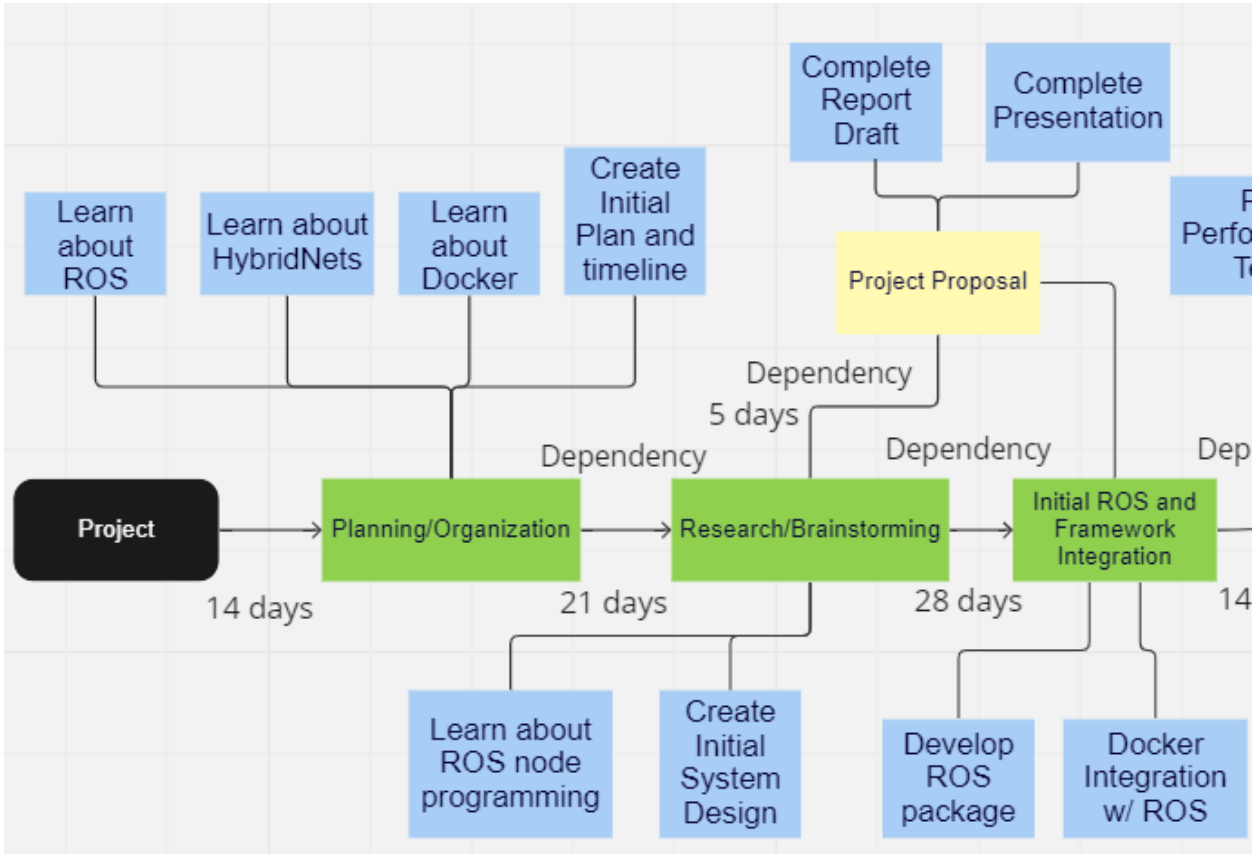



*Figure 13: PERT Chart Initial Stages*

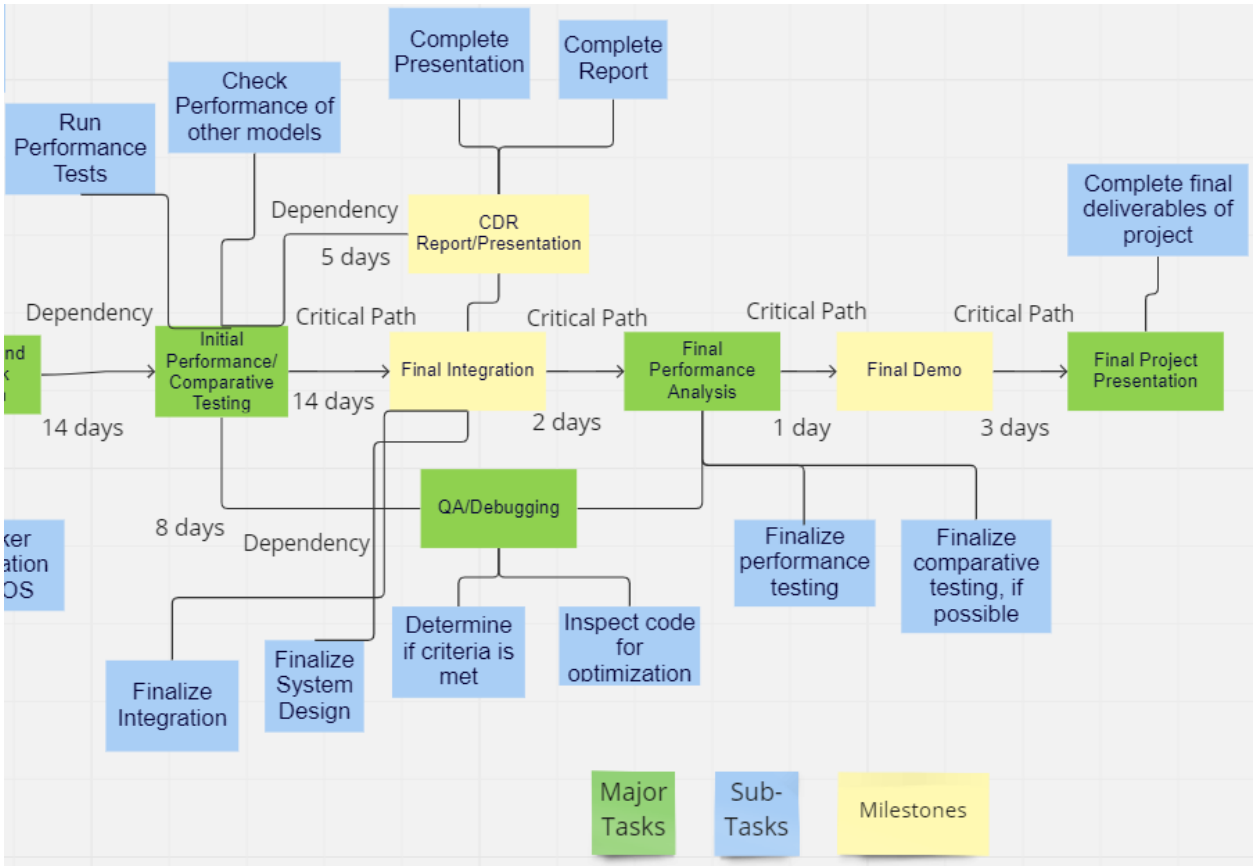



*Figure 14: PERT Chart Final Stages*

As seen in Figures 12-14, the main critical path includes the completion of the Initial ROS and Framework Integration, Initial Performance/Comparative Testing, Final Integration, Final Performance Analysis, Final Demo, and Final Presentation. The minimum time required to complete this project will take at least 62 days as suggested from our Gantt chart. The team has a total of 101 days to complete the project. The above Gantt chart depicts estimations of the time of completion for the project. The tasks on the Gantt chart have accounted for any roadblocks the team may encounter.

Regarding tasks, the team will meet every week to create a list of tasks that need to be completed for that week. Each task will be assigned to the person who best fits that role. As discussed in Section 5.1, the roles each member possesses will aid in assigning tasks. Each member is encouraged to collaborate together to collectively move forward. The team will closely monitor the tasks that are completed within the given timeframe and determine which would need to be pushed back if needed. Each member will be held accountable for their assigned tasks. The Gantt chart will be updated in terms of percentage completed when needed. The team will prioritize tasks based on the critical path of the project to ensure that proper progress is made. It is possible that requirements may change regarding the project and thus the team will remain flexible to account for any changes.

### 3.2 Updated validation and testing procedures

Our project has two main deliverables that we must validate to evaluate our overall success for meeting our goals. It is important to remember that this project would be a waste unless we do our due diligence and validate that we have improved upon the current system and didn't just reinvent the wheel. In order to

do this we will have to run a variety of tests. We will be testing each of these deliverables and individual objectives by comparing the inputs and outputs from ROS and the joint detection framework at every step of the way. If we are able to prove that we have accomplished our two main deliverables, it will show that we have proof of a working system that solves our need statement.

The first deliverable we must validate is making sure we have made robust and valuable comparisons between the current ROS pipeline that uses separate lane detection and object detection models and the external framework we were provided with that uses a combined lane and object detection model. One way we can make our comparisons is by simply using information and data provided by ROS and the external framework and simply compare their findings. Even though this could lead to accurate results, this is not the best approach and needs further steps to validate our results and existing data. We will make our own comparisons and validate these results by doing extensive testing using data samples on each framework. This will allow us to back-up our findings and validate our comparisons on the two frameworks on both ideal and non-ideal conditions. These tests we run will help us validate our predictions amd comparisons on accuracy and efficiency, which are the two main criteria we are trying to improve over the current ROS system. For the accuracy, the specific metrics we will be using are the accuracy percentages generated by the detection models that are shown above each object. These tell us how likely an object that is detected is a car or lane for instance. For the performance speed the specific metric we will be using is how many seconds it takes from starting the detection models to displaying the results to the screen. Just running the same data samples on each framework will not be enough to validate our results however. We need to run a large variety of datasets on each framework, including possibly samples included in ROS that are heavily documented, samples included in the external framework documentation, samples provided by the instructor and TA, and even samples we as a team find online or create ourselves using videos/images from streets around A&M.

The second main deliverable we must validate is that the new pipeline we built actually works as intended. This pipeline will use ROS nodes to take in images as an input, call the functions present in the external framework that uses a joint lane and object detection model, and return the new images with bounding boxes and classification results. These three steps can all be validated individually. We can validate whether the ROS nodes received the image by using ROS' existing functions to print out its nodes' input and verify it represents the image we passed in as an input. We can validate whether the external combined model framework has received the image using the same process, and also validate whether the external framework is actually reading through the image and trying to recognize lanes and objects by printing out outputs from those corresponding functions while we run the combined framework using the data sets. Finally, we can validate that the ROS nodes received the updated image with bounding boxes and classification results and that the model did perform as intended by comparing the image ROS receives at the end with the image resulting from running the external framework using the same image as input without ROS integrated. Validating all these steps will prove that we have indeed accomplished the objectives of our project and met our stated needs.

On top of all of this there are even more ways we can validate and test our model to prove it solves our need statement. We need to check to see how our accuracy and performance speed of the new system compares to that of the previous system, while ensuring the entire experience is still up to ROS standards in order to increase or maintain the number of users who want to use lane and object detection within ROS. Lasly we will also validate that the user experience is up to standards by having a variety of test-subjects use our new system. These test-subjects should have a range of experience with ROS and machine learning, from zero experience to years of experience. These test subjects can then give us feedback on something that can't be measured by strict math and equations like accuracy and performance speed. We then can use this feedback to improve upon our system to make it more seamless and and overall more enjoyable experience.

### 3.3 Updated division of labor and responsibilities

The team has split up into pairs to do pair programming sessions togethers. Jason Bernal and Pavan Poladi have looked into the HybridNets model while Khai and Viet have looked into ROS and Docker integration. The team chose to do this as Jason and Pavan would like to better understand Machine Learning models and Khai and Viet like to learn new technologies and work with Docker.

The pairs intend to meet at least once a week outside of normal class hours to do pair programming sessions. The pairs find a time that works for both of them due to their busy schedule as full time students. The team has mainly utilized Discord as a medium to discuss and collaborate on the project. Pair programming has increased the productivity of the team. The team initially had mob coding sessions, where all members attend a meeting, but the team found that any delays in the programming session would cause all members to halt whereas with pair programming sessions, worst case, only one member is halted.

The Gantt Chart has been updated to showcase the high-level due dates of the team. The team continues to rotate the responsibilities of the weekly report, as suggested by the syllabus. Doing so has helped each member of the team continue to be up-to-date with all aspects of the project. The team helps each other with the report by also discussing updates that can be used to list on the report.

The PERT Chart has also been updated to showcase the dependencies and critical paths that can be found regarding the progress of this project. The team works actively to ensure that the critical path is met as it is necessary to advance the project forward. Both the Gantt chart and PERT chart are used to demonstrate the high-level objectives that the team has put forward to ensure that the project is completed in a timely manner.

The team has been able to complete the initial stages of the project, as seen on Figure 11. Overcoming the initial stages of the project has helped the team gain a better understanding of what to work on and how to divide the workload evenly. The team ensures that the workload is evenly distributed so that each member has the opportunity to contribute to the project. The team has grown immensely throughout the course of the project and are expected to continue to grow. The team has learned how each other's strengths and weaknesses, and help complement each other. An example of the growth that the team encountered was the adoption of pair programming. The team initially worked in mob coding sessions but found it to be ineffective compared to pair programming. The team began working in pairs of team members which would complement each other's work style and help with workflow. Afterwards, the team as an increase in the productivity. It is important to note that due to the fact that the team worked in pairs, the commits made are shared among the pair. To ensure that everyone receives credit for their contributions, the team alternate in who creates the commit to ensure that those who view our repository can see the member's activity. Each individual member has worked on various aspects of the project and have learned a great deal of new skills.

Regarding responsibilities, the team sets deadlines and responsibilities on a weekly basis. The team unanimously decides the objectives of the week and who is responsible for the completion of those objectives. These mini-objectives are subject to change and sometimes shared among team members to help in productivity. Here is a brief list of deliverables the team has set for the following two weeks:

**10/24/2022-10/30/2022**

10/24:

- Jason Bernal, significant progress to be made on the Critical Design Review Report
- Pavan Poladi, significant progress to be made on the Critical Design Review Report
- Khai Nguyen, significant progress to be made on the Critical Design Review Presentation
- Viet Nguyen, significant progress to be made on the Critical Design Review Presentation

10/25:

- Khai Nguyen, contributions to be made on ROS + Python Package Development
- Viet Nguyen, contributions to be made on ROS + Python Package Development
- Jason Bernal, contributions to be made on ROS + HybridNets integration
- Pavan Poladi, contributions to be made on ROS + HybridNets integration

10/30:

- Team submission of Critical Design Review Report
- Team submission of  Critical Design Review Presentation
- Viet's submission of Weekly Report

**10/31/2022-11/06/2022**

10/31:

- Team Critical Design Review Presentation
- Khai Nguyen, complete initial ROS + Python Package Development
- Viet Nguyen, complete initial ROS + Python Package Development
    - Catkin Package and wrapping needed dependencies in it
- Jason Bernal, complete initial ROS + HybridNets integration
- Pavan Poladi, complete initial ROS + HybridNets integration

11/02:

- Jason Bernal, begin ROS + HybridNets comparison testing
- Pavan Poladi, begin ROS + HybridNets comparison testing
- Khai Nguyen, begin ROS + HybridNets comparison testing
- Viet Nguyen, begin ROS + HybridNets comparison testing
    - Catkin Package and wrapping needed dependencies in it

11/05:

- Khai's submission of Weekly Report
- Viet Nguyen, complete Catkin Package and wrapping needed dependencies in it

As mentioned, lower-level tasks are assigned at the beginning/end of the week and thus are subject to change. Tasks beyond the stated dates are yet to be determined. Some tasks may be achieved ahead of schedule or may need to be pushed back. The team communicated on a regular basis the status of the tasks to be completed to help determine where more resources need to be devoted to. For better or worse, we are flexible in pair programming where we spread and move things around to each other so that we can have a better overall understanding of what is actually going on. The team is heavy on ensuring there is a collaborative nature among its members and actively work well together. We have found that frequent communication and meetings can ensure everyone is on the same page.

## 4 Preliminary results

With the initial stages of the ROS integration, the team wanted to learn how to do basic operations in ROS and have a basic understanding of its architecture. The team was able to code a simple Python script which allows a gray rectangle to be displayed onto an image and have that image shown on the ROS GUI. This showed us that we were effectively able to understand the subscriber and publisher model well enough to the point where we can actually make updates to ROS. This means that we were ready to begin testing the HybridNets model and then integrate it into ROS. The code for this demo is attached below. Here are the steps we take in this script to achieve our result. We used the callback_Img(data) function to take in image data from ROS and transform it into a CV2 image in rgb8 format. By transforming it into a CV2 image we gain a lot of flexibility on what we can add to the image. We then draw a gray rectangle using various CV2 functions and specify its size, starting and ending points, color, and thickness. Next we add image headers, and publish the image. Finally we subscribe to the associated topic and publish to it, allowing us to have control over where the data comes and goes. Lastly, we use the rospy.spin() function to replicate essentially a for loop to loop through ROS image data and display the gray rectangle on all presented images. ROS uses a callback function so that whenever a specific event happens, in this case, our ROS node receives an image from the original camera, the callback function will run. This makes the callback function reactive, which is nice for saving resources. This goes the same way for the subscribers that other nodes may use from our node.

```
import sys
import cv2
import rospy
import numpy as np
from cv_bridge import CvBridge
from sensor_msgs.msg import Image

bridge = CvBridge()
def callback_Img(data):
    img = bridge.imgmsg_to_cv2(data, desired_encoding='rgb8')
    # Start coordinate, here (5, 5)
    # represents the top left corner of rectangle
    start_point = (5, 5)

    # Ending coordinate, here (220, 220)
    # represents the bottom right corner of rectangle
    end_point = (220, 220)

    # Blue color in BGR
    color = (255, 0, 0)

    # Line thickness of 2 px
    thickness = 2
    img = cv2.rectangle(img, start_point, end_point, color, thickness)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    grayImageMsg = CvBridge().cv2_to_imgmsg(gray.astype(np.uint8))
    grayImageMsg.header = data.header
    grayImageMsg.encoding = '8UC1'
    grayImgPub.publish(grayImageMsg)

rospy.init_node('img_record_node')
rospy.Subscriber("/front_camera/image_raw", Image, callback_Img)
grayImgPub = rospy.Publisher('/img_gray', Image, queue_size=10)
rospy.spin()
```

*Figure 15: ROS code adding gray rectangle to images*

Our team has also been working hard to find out the best way to integrate the Hybrid Nets model into ROS, which is one of our primary objectives for this project. We first spent a lot of time looking through the very large Hybrid Nets code base and trying to figure out how all the pieces fit together. We ended up being able to get a demo working using the Hybrid Nets mode, however we realized that it simply won't be feasible to combine the entire codebase of Hybrid Nets into the ROS code base. This is because of the multitude of dependency errors that would arise as well as the fact that the system would just become too complicated and not allow for a good experience for the users or creators for ROS. We had to reconvene and think of a better strategy. We found that a better way would be to see if we can somehow import HybridNets into ROS using a series of simple import and configuration lines of code. After some searching on Google we found that Pytorch actually allows this. In order to achieve this you must install pytorch, import torch, and retrieve the HybridNets model, and designate whether you would like it to be pre-trained. Finally you can take an image and pass it through the model to retrieve the various results like the features, regression, classification, and more.

```python
import torch

# load model
model = torch.hub.load('datvuthanh/hybridnets', 'hybridnets', pretrained=True)

#inference
img = torch.randn(1,3,640,384)
features, regression, classification, anchors, segmentation = model(img)
```

*Figure 16: Basic script for importing and running HybridNets model*

The issue we had however is that this gave us the various key parts we needed, but we did not know how to display them onto the image so that the classification and bounding boxes are displayed in real time. This took us a lot of time searching online and testing out different things as we found the HybridNets documentation to be pretty poor in our opinion. However, we finally were able to figure this out and were able to have a functioning model working on various images using HybridNets. Below is the code and resulting images showing that the model is working as intended. This is a great step in our project, because now this means we are ready to essentially transfer this code we created using HybridNets into ROS and then figure out how to display the results onto ROS. It seems like there can also be complications in getting the Pytorch dependencies or certain dependencies may not allowed the torch hub to load the model in at all.

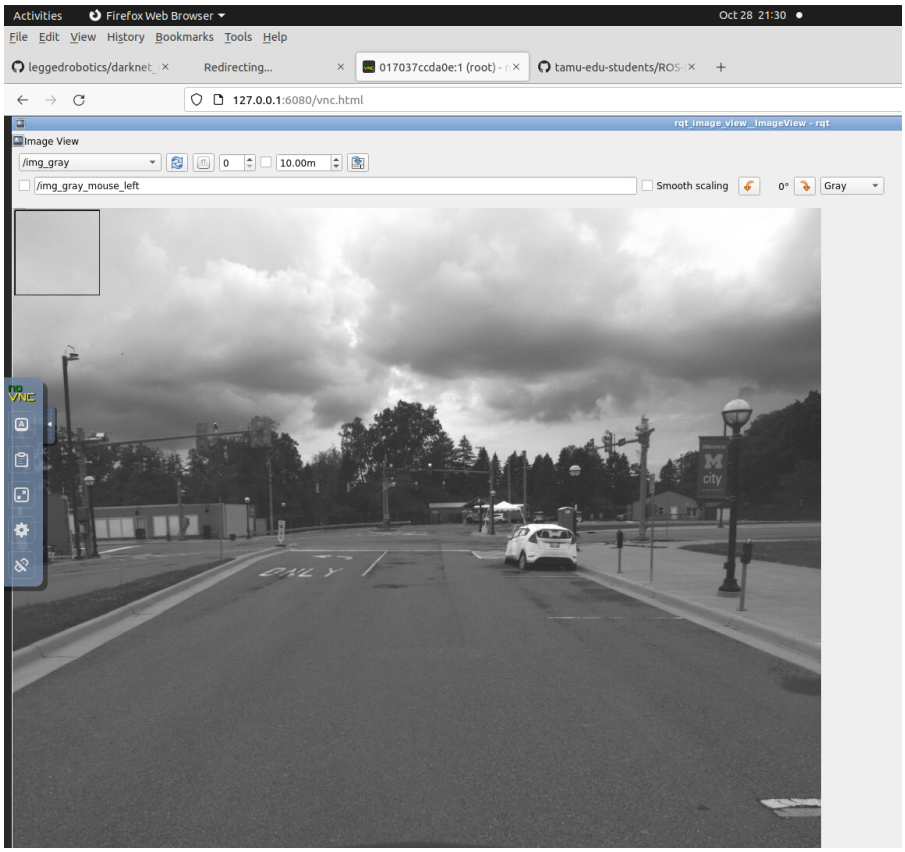



*Figure 17: Draw basic shapes onto ROS*

*Figure 18: Local HybridNets Test*

*Figure 19: Local HybridNets Test on ROS*

Finally, we also want to note that we have been working hard on creating a package that will allow users to run our HybridNets models in ROS and download and install all necessary dependencies with just a few command line arguments. The reason we created this is because our team has spent many hours over many days trying to figure out which specific configurations are needed to run HybridNets and ROS

together. We are still trying to figure this out ourselves as we run into new problems everyday. After the completion of our project we want new users to have a quick and seamless process when using our system. Unfortunately, we don't have this working as intended at the moment, due to various bugs we are facing, but we plan to have this completed by the end of the project period. This involves catkin workspaces, and trying to have a package so that other groups trying to use this node does not have quite the problems that we did trying to resolve dependencies.