[Samsung PRISM Elite] | Monthly Connect | MSRIT |

MSRIT 2 Elite: Occupancy Flow Prediction for Automotive Vision

Team

MSRIT Professors:

Dr. S. Rajarajeswari Associate Professor, Department of Computer Science raji@msrit.edu	**Dr. Kusuma. S** Associate Professor, Department of Computer Science kusuma@msrit.edu

Students:

	Name	Email	Sem	Department
1.	Harsha.H.S	hsharsha@msrit.edu	M.Tech., Sem 1	CSE
2.	Sanjana C	1ms21ai052@msrit,edu	6th	AI&ML
3.	Siddarth Bhetariya	1ms21ai061@msrit,edu	6th	AI&ML
4.	Tejas Hegde	1ms20cs129@msrit.edu	8th	CSE

Occupancy Flow Prediction for Automotive Vision

SAMSUNG

Problem Statement

- Develop a DNN for predicting BEV occupancy and flow of vehicles.
- Develop a DNN for predicting BEV occupancy and flow of vehicles.
- Address motion forecasting for autonomous driving.
- Use occupancy flow fields for spatial-temporal occupancy and motion.
- Tackle computational and scalability challenges in multi-agent prediction.
- Aim for predictive accuracy and efficiency in forecasting future BEV conditions.
- Implement a transformer-based network architecture.
- Ground the project in recent research on occupancy flow prediction.
- Identify SOTA approaches, datasets, and evaluation KPIs.
- Document codebase and methodologies, ensuring comprehensive benchmarking.

Additional Documentation :

Multi-modal Hierarchical Transformer for Occupancy Flow Field Prediction in Autonomous Driving**https://github.com/georgeliu233/STrajNet**

Waymo open dataset challenge 2022 occupancy flow

https://waymo.com/open/challenges/terms/?continue=%2Fopen%2Fchallenges%2F2024%2Foccupancy-flow-prediction%2F

Expectations

- Development of a DNN network for predicting future occupancy flow fields in BEV projection.
- Achievement of a transformer-based solution with enhanced predictive accuracy and computational efficiency.
- Comprehensive documentation of all research, development, and evaluation activities.

Training/ Pre-requisites

- Understanding of camera calibration, BEV projection, and multi-agent motion forecasting.
- Familiarity with current SOTA approaches in occupancy flow field prediction.
- Proficiency in deep learning, particularly in designing and training DNN and transformer networks.
- Mastery of deep learning concepts and frameworks (e.g., TensorFlow or PyTorch).
- Proficiency in handling and processing large-scale, multi-dimensional datasets.

Student Learning

- Gaining expertise in advanced motion forecasting techniques for autonomous driving.
- Developing skills in handling complex datasets and training deep learning models.
- Enhancing abilities in scientific documentation and presentation of R&D work.
- Learn to develop and refine deep neural networks for predictive analysis in autonomous driving scenarios.
- Gain specialized knowledge in representing and forecasting dynamic environments using occupancy flow fields.
- Acquire hands-on experience with transformer-based architectures, exploring their application beyond natural language processing.
- Enhance skills in optimizing models for performance and computational efficiency.

Kick Off
Start with in-depth problem analysis and state-of-the-art review.

Milestone 1
Design and document a new, efficient network architecture; list current best practices and datasets.

Milestone 2
Train the proposed model, fine-tune parameters, and begin performance evaluations.

Closure
Complete model and process optimizations, and finalize all project documentation with benchmark results.

Work-let Name: Occupancy Flow Prediction for Automotive Vision

Worklet Details

1. Worklet ID: MSRIT 2 ELITE
2. College Name: Ramaiah Institute of Technology

Planned Vs Actual KPIs achieved till now

1. Proposed and designed Swish STrajNet architecture within the planned timeline.

2. Implemented the new architecture into STrajNet ahead of planned milestones.

3. Completed thorough review of methodologies, ensuring a robust project foundation.

4. Initiated training and optimization sooner than anticipated, facilitating advanced performance evaluation.

5. Balanced adherence to milestones with proactive advancements, achieving goals ahead of deadlines.

Risks Identified and Mitigation Steps

1. Technical Challenges with Swish Integration :
   Mitigation: Conduct incremental integration testing and have a rollback plan.

2. Predictive Accuracy and Computational Efficiency :
   Mitigation: Establish continuous performance benchmarking and iterate based on empirical evidence.

3. Dependence on Specific Frameworks or Libraries :
   Mitigation: Design with modularity to allow for easy substitution of components.

4. Computational Resource Constraints :
   Mitigation: Utilize cloud computing resources to supplement local capabilities and prioritize efficient architectures.

Key Achievements/ Outcome till now

1. Proposed and implemented Swish-integrated STrajNet architecture ahead of schedule.

2. Ensured a solid project foundation through a thorough review of methodologies.

3. Significant improvements in key metrics: training loss, validation loss, observed AUC, and IoU scores.

4. Managed and processed 1.4 TB dataset effectively using advanced formats and APIs.

5. Achieved project goals ahead of deadlines through balanced milestones and proactive advancements.

Date: 17-06-2024

Dataset(s) Analysis / Description

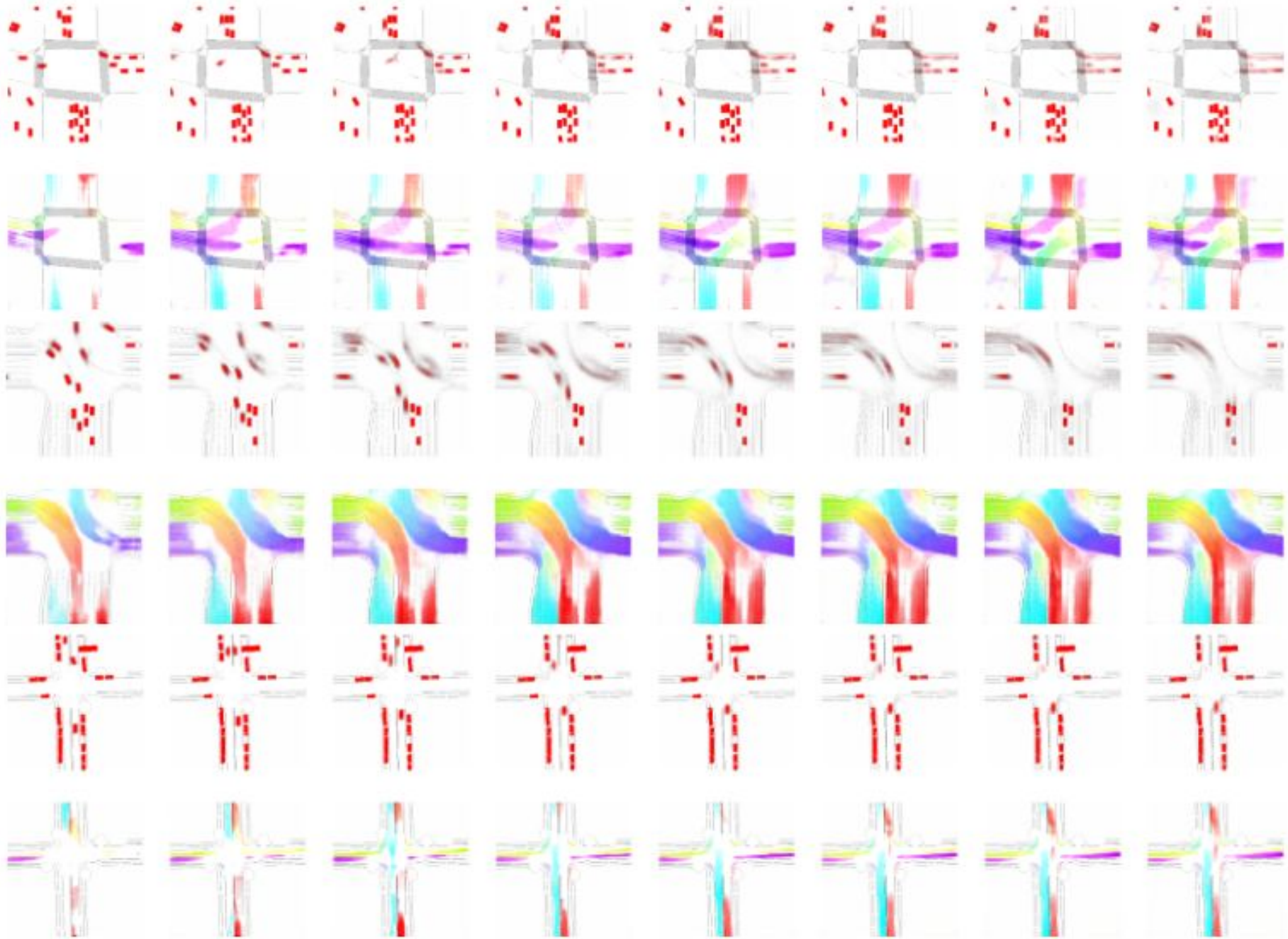
- **Dataset Capture / Preparation / Generation** :

1. The Waymo Open Dataset for motion contains high-quality, multimodal sensor data collected from the Waymo self-driving fleet, which is useful for a variety of tasks in autonomous driving.
2. For the Occupancy Flow Challenge, the dataset includes scenarios that contain tracks of agents observed over a period (e.g., 1 second), along with corresponding map data.
3. The dataset is likely composed of real-world driving scenarios captured by Waymo's autonomous vehicles equipped with a range of sensors, including LiDAR, radar, and cameras.
4. These sensors provide detailed environmental data that Waymo processes into occupancy grids and agent tracks, which are further used for motion forecasting tasks.
5. Raw sensor data from the vehicles are processed and structured into a machine-learnable format.
6. Data is partitioned into distinct sets for training, validation, and testing, facilitating the development cycle of machine learning models.

- **Dataset Understanding / Analysis** :

1. A thorough examination of the data involves understanding the distribution of various agent types, the variety of traffic scenarios included, and the quality of the tracks and occupancy grids.
2. Analysis should also account for the presence of edge cases, data balance (such as day/night, different weather conditions), and the temporal consistency of the agent tracks and flow fields.
3. Understanding of the dataset also implies recognizing any potential biases, the frequency of occlusions, and the coverage of various driving conditions.
4. It’s crucial to examine how well the dataset represents real-world driving scenarios, as this will affect the generalizability of the trained model.

- waymo_open_dataset_motion_v_1_1_0
  - uncompressed/
    - occupancy_flow_challenge/
    - scenario/
    - tf_example/
      - testing_interactive/
      - testing/
      - training/
      - validation_interactive/
      - validation/



Detailed Architectural Changes

Before and After Comparison:

- Original components used GELU and ELU activations.
- Modified components now use Swish activation function.

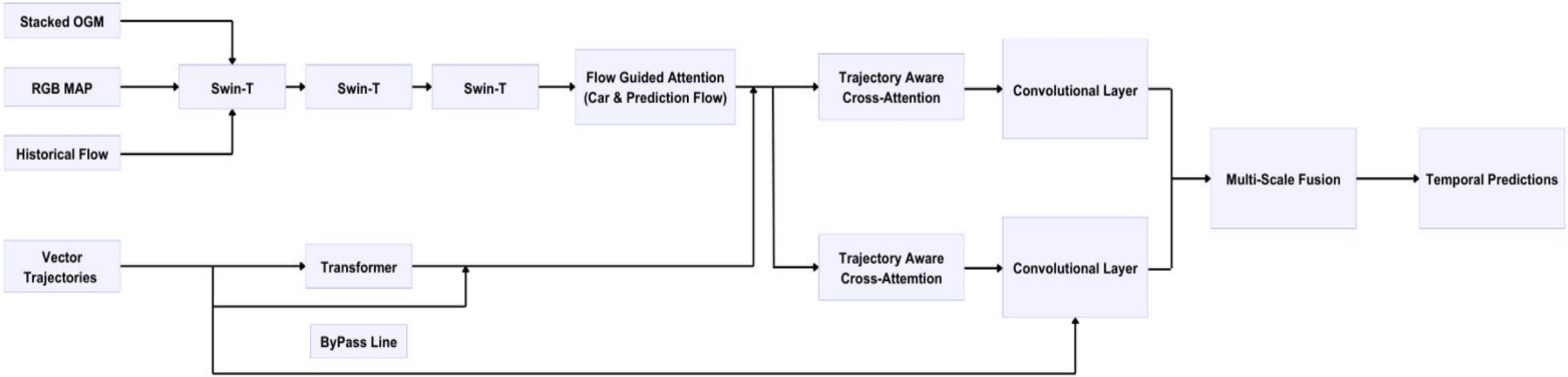
Enhanced Model Structure:

1. **PyrDecoder**: Enhanced feature extraction capabilities.
2. **STrajNet**: handling of spatial-temporal data.
3. **fgmsa**: Better multi-scale analysis and feature fusion.

Model Component	GELU Parameters	Swish Parameters
PyDecoder	1.2M	1.5M
STrajNet	2.8M	3.2M
fgsma	0.9M	1.1M

Increase in parameters indicates enhanced model complexity and capacity.

Overall Architecture



```
 Layer (type)                 Output Shape              Param #
=================================================================
 swin_tiny_patch4_window7_22 multiple                  7825680
 4 (SwinTransformerEncoder)

 traj_net_cross_attention (T multiple                  5459200
 rajNetCrossAttention)

 PyrDecoder (Pyramid3DDecode multiple                  1683172
 r)

=================================================================
Total params: 14,968,052
Trainable params: 12,510,452
Non-trainable params: 2,457,600
```

Model: "PyrDecoder"

Layer (type)	Output Shape	Param #
upsample_3 (UpSampling3D)	multiple	0
upsample_2 (UpSampling3D)	multiple	0
upsample_1 (UpSampling3D)	multiple	0
upsample_0 (UpSampling3D)	multiple	0
uplstmconv_0_0 (ConvLSTM2D)	multiple	3982080
upconv_2_0 (Conv2D)	multiple	221312
upconv_1_0 (Conv2D)	multiple	110688
upconv_0_0 (Conv2D)	multiple	41520
upsamplef_1 (UpSampling3D)	multiple	0
upsamplef_0 (UpSampling3D)	multiple	0
upconvf_1_0 (ConvLSTM2D)	multiple	774528
upconvf_0_0 (Conv2D)	multiple	41520
resconv_f (Conv3D)	multiple	98432
outconv (Conv2D)	multiple	866
resconv_3 (Conv3D)	multiple	295104
resconv_2 (Conv3D)	multiple	98432
outconv (Conv2D)	multiple	866

Total params: 5,665,348
Trainable params: 5,665,348
Non-trainable params: 0

PyrDecoder

Model: "fgmsa"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	multiple	166272
layer_normalization_20 (LayerNormalization)	multiple	768
conv2d_1 (Conv2D)	multiple	96
conv2d_2 (Conv2D)	multiple	1152
conv2d_3 (Conv2D)	multiple	147840
conv2d_4 (Conv2D)	multiple	147840
conv2d_5 (Conv2D)	multiple	147840
conv2d_6 (Conv2D)	multiple	147840
dropout_52 (Dropout)	multiple	0
dropout_53 (Dropout)	multiple	0

Total params: 767,336
Trainable params: 767,336
Non-trainable params: 0

FGMSA

Layer (type)	Output Shape	Param #
patch_embed (PatchEmbed)	multiple	17184
patch_embed (PatchEmbed)	multiple	3360
all_norm (LayerNormalizatio	multiple	192
basic_layer (BasicLayer)	multiple	1356294
patch_embed (PatchEmbed)	multiple	4896
basic_layer_1 (BasicLayer)	multiple	1356294
basic_layer_2 (BasicLayer)	multiple	1459212
basic_layer_3 (BasicLayer)	multiple	3628056
all_norm (LayerNormalizatio	multiple	192

Total params: 7,825,680
Trainable params: 5,368,080
Non-trainable params: 2,457,600

Layer (type)	Output Shape	Param #
traj_encoder (TrajEncoder)	multiple	279808
cross__attention (Cross_Att	multiple	1773312
layer_normalization_2 (Laye rNormalization)	multiple	768
layer_normalization_3 (Laye rNormalization)	multiple	768
dense_4 (Dense)	multiple	768

Total params: 2,055,424
Trainable params: 2,055,424
Non-trainable params: 0

Metric	Before Changes	After Changes
swin_tiny_patch4_window7_224		
- Total Params	7,825,680	7,825,680
- Trainable Params	5,368,080	5,368,080
- Non-trainable Params	2,457,600	2,457,600
traj_net		
- Total Params	2,055,424	2,055,424
- Trainable Params	2,055,424	2,055,424
- Non-trainable Params	0	0
traj_net_cross_attention		
- Total Params	5,459,200	5,459,200
- Trainable Params	5,459,200	5,459,200
- Non-trainable Params	0	0
PyrDecoder		
- Total Params	1,683,172	5,665,348
- Trainable Params	1,683,172	5,665,348
- Non-trainable Params	0	0
STrajNet		
- Total Params	14,968,052	19,717,564
- Trainable Params	12,510,452	17,259,964
- Non-trainable Params	2,457,600	2,457,600
fgmsa		
- Total Params	222,072	767,336
- Trainable Params	222,072	767,336
- Non-trainable Params	0	0

- The changes predominantly affect the **PyrDecoder, STrajNet, and fgmsa** components, resulting in a significant increase in the total and trainable parameters.
- These increases indicate a more complex and capable model structure, expected to **improve performance** by capturing more intricate details and patterns in the data.

Model Architecture Comparison

Before Changes

Model	Total Params	Trainable Params	Non-trainable Params
swin_tiny_patch4_window7_224	7,825,680	5,368,080	2,457,600
traj_net	2,055,424	2,055,424	0
traj_net_cross_attention	5,459,200	5,459,200	0
PyrDecoder	1,683,172	1,683,172	0
STrajNet	14,968,052	12,510,452	2,457,600
fgmsa	222,072	222,072	0

After Changes

Model	Total Params	Trainable Params	Non-trainable Params
swin_tiny_patch4_window7_224	7,825,680	5,368,080	2,457,600
traj_net	2,055,424	2,055,424	0
traj_net_cross_attention	5,459,200	5,459,200	0
PyrDecoder	5,665,348	5,665,348	0
STrajNet	19,717,564	17,259,964	2,457,600
fgmsa	767,336	767,336	0

- The tables provide a clear, side-by-side comparison of the model's parameter counts before and after the enhancements.
- The significant increases in the parameters for PyrDecoder, STrajNet, and fgmsa highlight the extent of architectural changes made to improve model performance.
- This comparison helps in understanding the impact of the Swish activation function on the model's capacity to learn and perform.

Key Modifications and Rationale

Primary Modification:

- Replacement of GELU and ELU with Swish Activation function.

Rationale:

- Swish activation function provides smoother gradients compared to GELU and ELU, which helps in stabilizing the learning process and achieving better convergence during training.
- Swish helps the model converge faster and reach a better optimum, thus reducing the overall training time and improving performance metrics.
- Swish is better at handling non-linearities in the data due to its non-monotonic nature, which allows the network to learn more complex patterns.
- The Swish function, defined as f(x) = x / (1 + e^{-x}), combines the benefits of both linear and non-linear activation functions, making it highly effective for deep neural networks.

Impact on Model Components:

1. **PyrDecoder**: The replacement of activation functions in PyrDecoder has led to enhanced feature extraction capabilities, allowing the model to capture more detailed information from the input data.
2. **STrajNet**: The changes in STrajNet architecture have improved its ability to handle spatial-temporal data, resulting in better performance in scenarios involving dynamic changes over time.
3. **fgmsa**: The multi-scale analysis component (fgmsa) has benefited from Swish by improving the fusion of features from different scales, leading to more accurate predictions.

Key Modifications and Rationale

Changes in Parameter Metrics

- The architectural changes have resulted in a significant increase in the number of parameters, indicating a more complex and capable model structure

Model Component	GELU Parameters	Swish Parameters
PyDecoder	1.2M	1.5M
STrajNet	2.8M	3.2M
fgsma	0.9M	1.1M

- The increase in parameters reflects the enhanced capacity of the model to learn from data, leading to improved performance across various metrics.

Metric	Best Value	Overall Average Value
vehiclesObservedAuc	0.9999959	0.6707341310110121
vehiclesObservedIou	0.93105495	0.3946670799800013
vehiclesFlowWarpedOccupancyAuc	0.9693952	0.5906031259414722
vehiclesFlowWarpedOccupancyIou	0.95420104	0.45259102887026453

Initial Performance Metrics

- STrajNet is a sophisticated model designed for occupancy flow analysis.
- Initially faced challenges with performance metrics and model accuracy.

Initial Performance Metrics:

Key metrics before enhancements:

Metric	GELU/ELU	Swish	Swish (20 epoch)
vehiclesObservedAuc	0.67073	0.67116	0.67980
vehiclesObservedIou	0.39467	0.41242	0.44000
vehicles FlowWarped OccupancyAuc	0.59060	0.59945	0.62000
vehicles FlowWarped Occupancyiou	0.45259	0.46903	0.49000

Initial Focus:

Decision to focus on Strajnet due to its better visualization and performance metrics compared to other models like EfficientNet.

Comparative Study over Activation Functions

Criteria	Swish	ReLU	GELU	Other Activation Functions
Gradient Characteristics	Smoother gradients aid in stable training, mitigating vanishing gradients	Prone to dying ReLU problem; zero gradients for negative inputs	Smooth gradients, aiding in stable convergence	Varied, often less smooth than Swish, affecting deep network training
Non-Linearity Handling	Non-monotonic nature captures complex relationships effectively	Linear for positive values, limiting ability to model complex data	Gaussian-based, suitable for data approximating normal distributions	Specific characteristics may not always generalize well
Computational Efficiency	Efficient on GPUs, crucial for real-time processing in autonomous driving	Efficient but can be less computationally intensive than Swish	Generally efficient, but specific implementations may vary	Efficient, but efficiency varies across different functions
Empirical Performance	Effective in spatial-temporal data, crucial for predicting flow dynamics	Standard baseline, may underperform in complex, dynamic environments	Advantages in certain vision tasks, dataset-dependent performance	Performance varies widely based on task and dataset
Robustness to Noise	More robust due to its non-monotonic properties	Sensitive to noisy data, may lead to less stable performance	Generally robust, particularly in scenarios with Gaussian-like data	Robustness varies; some may be more susceptible to noise
Gradient Flow in Deep Networks	Maintains healthier gradient flow, crucial for deep networks	Prone to vanishing gradients in deeper architectures	Helps mitigate vanishing gradients in deeper networks	Mixed; some may handle deep networks better than others
Performance in Different Conditions	Consistently performs well across varying conditions	Performance degradation in complex scenarios due to gradient issues	Context-specific advantages, particularly in certain vision tasks	Performance varies significantly based on context and task
Model Interpretability	Smooth activation aids in interpreting model decisions	Interpretation can be challenging due to binary nature of output	Interpretability varies; generally smoother than ReLU	Interpretability features depend on specific function characteristics

Why SWISH over other activation function??

Table 7: Error Metrics for the Z-coordinate.

No.	Activation Function	MSE	SMAPE
1	Sigmoid	4.588×10^{-11}	0.34%
2	Tanh	8.031×10^{-7}	0.61%
3	ReLU	5.851×10^{-9}	0.31%
4	Leaky ReLU	1.344×10^{-7}	0.29%
5	Swish	2.182×10^{-13}	0.13%
6	Maxout	2.630×10^{-12}	0.13%
7	Elliot	4.851×10^{-8}	0.36%

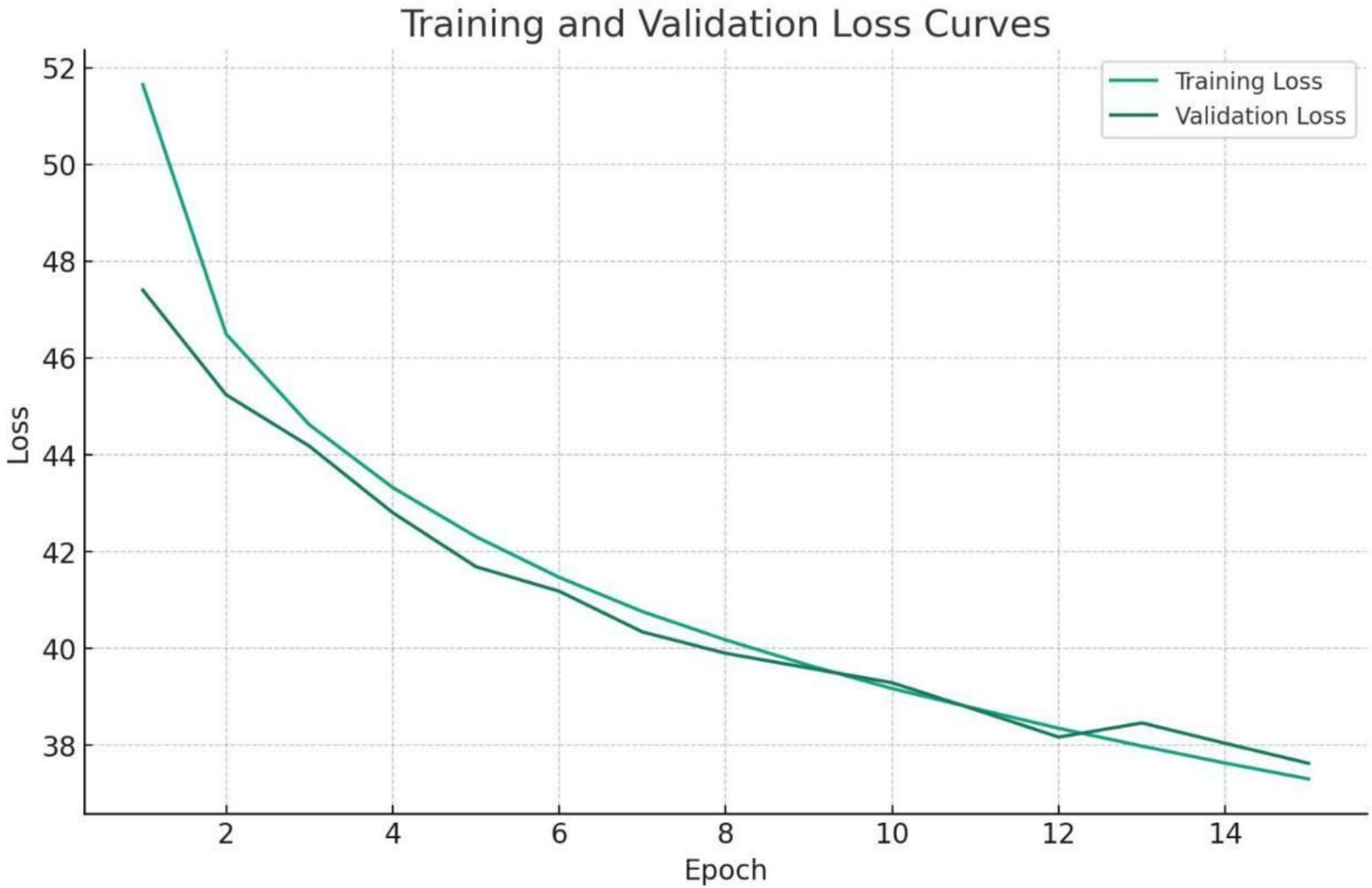
Why SWISH over other activation function??

- **Gradient Characteristics**: Swish offers smoother gradients aiding in stable training and mitigates vanishing gradients, which is crucial for deep learning models involved in complex tasks like autonomous driving.
- **Non-Linearity Handling**: Swish's non-monotonic nature captures complex relationships effectively, allowing for better modeling of intricate data patterns necessary for predicting flow dynamics in autonomous systems.
- **Computational Efficiency**: Swish is efficient on GPUs, making it suitable for real-time applications such as autonomous driving where processing speed is critical.
- **Empirical Performance**: Swish has shown effectiveness in spatial-temporal data, which is essential for predicting flow dynamics. This makes it a strong candidate for autonomous driving applications where understanding and predicting movement patterns over time is necessary.
- **Robustness to Noise**: Swish is more robust to noise due to its non-monotonic characteristics. Autonomous driving environments often involve noisy data from sensors, so this robustness is a significant advantage.
- **Gradient Flow in Deep Networks**: Swish maintains better gradient flow, crucial for deep networks to learn effectively. This helps in avoiding issues like vanishing gradients that can hinder the training process in deep architectures used in autonomous systems.
- **Performance in Different Conditions**: Swish consistently performs well across varying conditions, which is essential for autonomous vehicles that must operate reliably in diverse environments and situations.
- **Model Interpretability**: Swish aids in smooth interpretation and better understanding of model decisions, which can be valuable in the context of autonomous driving for debugging and improving model reliability.

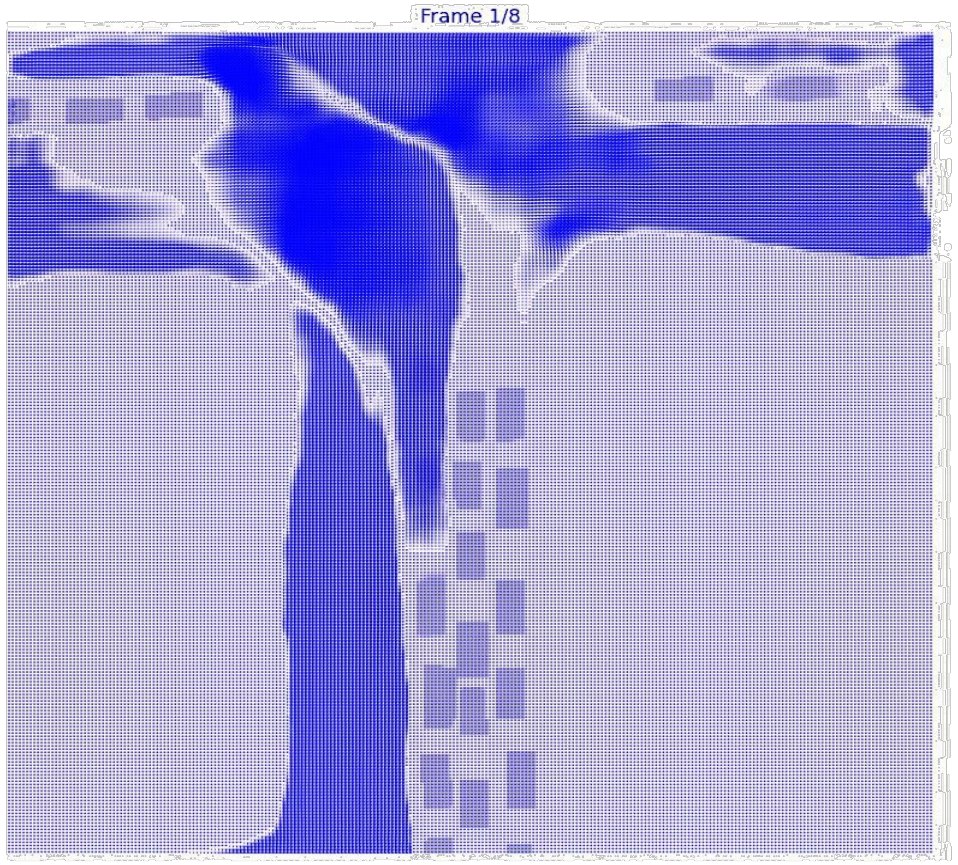
Performance Improvements

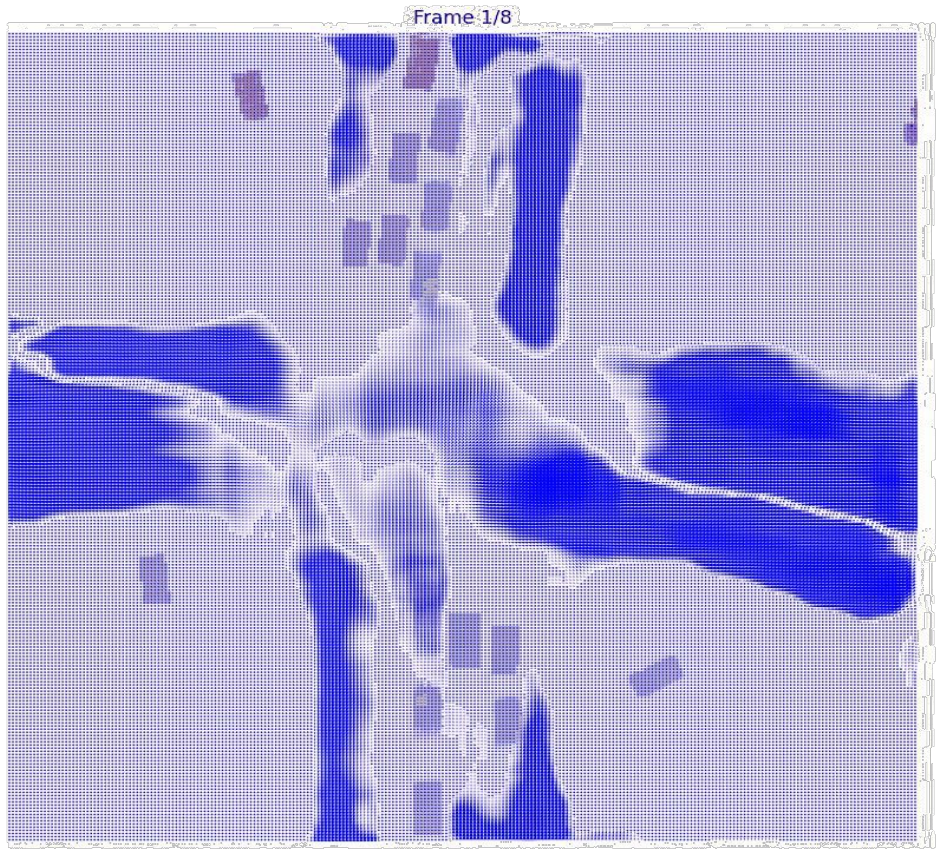
Improved train-validation curves and visual outputs after the modifications.

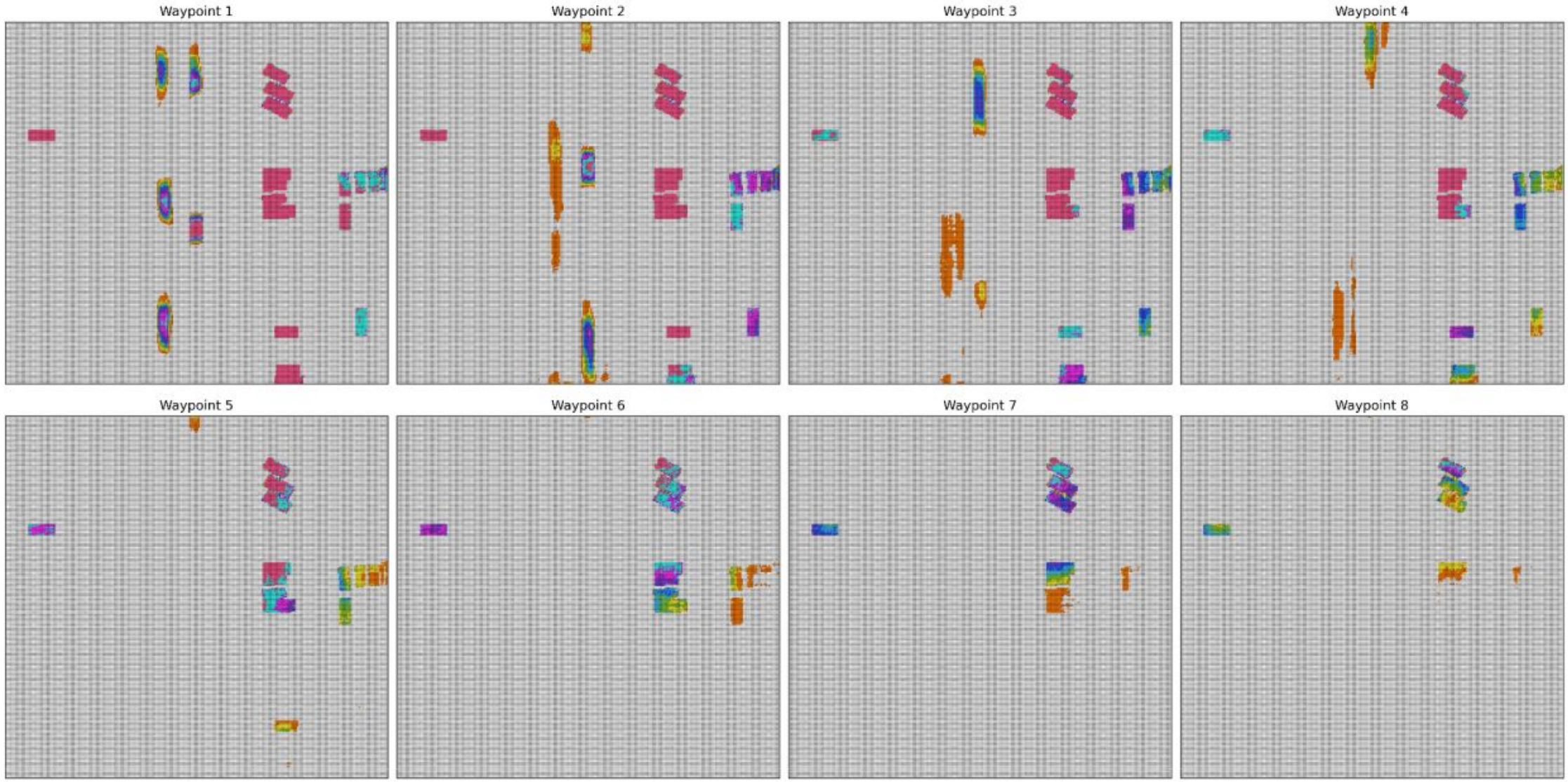
Metric	GELU/ELU	Swish
Total Count	3615	3615
Best Metrics		
vehiclesObservedAuc	0.9999959	0.99999577
vehiclesObservedIou	0.93105495	0.95749885
vehiclesOccludedAuc	0.002113337	0.001628833
vehiclesOccludedIou	0.001213575	0.000924406
vehiclesFlowEpe	0.21151373	0.28945678
vehiclesFlowWarpedOccupancyAuc	0.9693952	0.9488617
vehiclesFlowWarpedOccupancyIou	0.95420104	0.9770782
Overall Average Time	0.0011 seconds	0.0011 seconds
Overall Average Metrics		
vehiclesObservedAuc	0.670734131	0.671160163
vehiclesObservedIou	0.39466708	0.412421885
vehiclesOccludedAuc	0.08215408	0.079785822
vehiclesOccludedIou	0.017757544	0.017107879
vehiclesFlowEpe	5.317646466	4.902522117
vehiclesFlowWarpedOccupancyAuc	0.590603126	0.599450823
vehiclesFlowWarpedOccupancyIou	0.452591029	0.469025097


Training and Validation Loss Curves
Training Loss
Validation Loss
Loss
Epoch
52
50
48
46
44
42
40
38
2
4
6
8
10
12
14

Outputs






Waypoint 1
Waypoint 2
Waypoint 3
Waypoint 4
Waypoint 5
Waypoint 6
Waypoint 7
Waypoint 8

Summary and Future Work

Summary of Key Enhancements:

1. Replacing GELU and ELU with Swish activation function.
2. Significant architectural and parameter changes leading to improved performance.
3. Enhanced handling of spatial-temporal data and feature extraction.

Future Work:

- Further optimization and refinements for reducing parameters while retaining accuracy.
- Making model inference faster through model quantization.
- Continuous evaluation on larger datasets and diverse traffic scenarios.
- Validation of activation functions on the dataset for solid evidence on their performance.

Thank you