UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Robust Visual Odometry for Realistic PointGoal Navigation

Author:
Ruslan PARTSEY

Supervisor:
Oleksandr MAKSYMETS

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Ruslan PARTSEY, declare that this thesis titled, "Robust Visual Odometry for Realistic PointGoal Navigation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Robust Visual Odometry for Realistic PointGoal Navigation

by Ruslan PARTSEY

Abstract

The ability to navigate in complex environments is a fundamental skill of a home robot. Despite extensive study, indoor navigation in unseen environments under noisy actuation and sensing and without access to precise localization continues to be an open frontier for research in Embodied AI. In this work, we focus on designing a visual odometry module for robust egomotion estimation and it's integration with navigation policy for efficient navigation under noisy actuation and sensing. Specifically, we study how the observations transformations and incorporating meta-information available to the navigation agent impacts visual odometry model generalization performance. We present a set of regularization techniques that can be implemented as train- and test-time augmentations to increase the robustness to noise.

Navigation agent, equipped with our visual odometry module, reaches the goal in 86% of episodes and scores 0.66 SPL in Habitat Challenge 2021 benchmark.

Acknowledgements

It was a great honor for me to work under the supervision of Oleksandr Maksymets. His expertise in Embodiend AI and PointGoal navigation was invaluable in formulating the research questions and methodology. I would also like to thank Oleksandr for organizing a Zoom meetings with the co-authors of state-of-the-art approaches in PointGoal navigation.

I appreciate the work that was done by our program director Olexii Molchanovskyi and his team to ensure that the research process goes smooth and the only thing students have to care about is studying.

Also, I would like to acknowledge Rostyslav Hryniv and Oles Dobosevych for consultations in Linear Algebra and Deep Learning related topics and providing a feedback on my research ideas and hypotheses.

All experiments were run using computational resources generously provided by Ukrainian Catholic University Machine Learning Lab [1].

And, of course, many thanks to the Grammarly [2] for covering 50% of my tuition fees.

[1]https://apps.ucu.edu.ua/en/mllab
[2]https://www.grammarly.com

Contents

List of Figures

List of Tables

To my mother Lesya Partsey. Without her support and help I wouldn’t be who I am today.

Chapter 1

Introduction

1.1 Motivation

With advances in science and engineering, technologies that were expensive and used only by large corporations or government departments (in most cases Department of Defense) became cheaper and "more affordable" and move from laboratories to our houses. Even though current progress in autonomous robotics is distant from how it is described in science fiction, Embodied AI is already being used in many different applications. In particular, the latest versions of the Roomba [1] robot vacuum cleaner use suite of sensors and visual localization system to learn and remember where objects are in a room, so that the robot knows exactly where it is, where it has been and where it needs to go next. But Embodied AI tasks extend far beyond exploration [2], for instance, gripping and moving objects to specified locations or natural language instructions following. The potential benefits of AI assistants of the future range from conveniences, such as asking a robot to get a set of keys from the kitchen to helping the visually impaired navigate unfamiliar environments or perform complex tasks in dangerous or difficult situations.

As a result, the study of intelligent systems with a physical or virtual embodiment (Embodied AI) is getting more and more attention from the research community nowadays.

Growing number of scientific papers published and competitions/workshops held at top Computer Vision / Machine Learning conferences (Wijmans et al., 2020; Chaplot et al., 2020; Ramakrishnan, Al-Halah, and Grauman, 2020; Datta et al., 2020) indicate the interest in the Embodied AI domain.

Research community has introduced wide range of navigation tasks (Anderson et al., 2018) so far. Proposed tasks can be distinguished by the type of goal:

- PointGoal - the agent must navigate to a specific location;
- ObjectGoal - the agent must navigate to a given object class;
- AreaGoal - the agent must navigate to an area of a specified category.

PointGoal navigation is considered to be investigated the most. It serves as a starting point in the Embodied AI navigation research and the developed approaches can be transferred to other navigation tasks.

The important question that is raised in recent publications is the ability of intelligent system to efficiently navigate in real-world environments (Kadian et al., 2019; Datta et al., 2020). Since launched in the house robot has to avoid/step over stairs,

[1]Roomba is a series of autonomous robotic vacuum cleaners sold by iRobot.

[2]Exploration is an Embodied AI task where the objective is to maximize the coverage in a fixed time budget.

deal with obstacles/clutter and different types of surface properties and lightning conditions. The focus is on increased realism i.e. experimental setups are designed to match the real world environment as close as possible (Kadian et al., 2019). That is achieved by incorporating actuation and sensing noise models.

Among the different sensor modalities, cameras are cheap and provide rich information of the environment that allows for robust and accurate place recognition. Moreover, compass and GPS data can be too noisy or simply unavailable in indoor spaces. Therefore, visual navigation solutions where the main sensor is a RGB-D camera are of major interest nowadays.

Significant progress in the visual navigation has been made by the researchers so far. Learning based approaches, consisting of Reinforcement Learning (RL) navigation policy and visual odometry module that is used for online pose estimation can learn the navigation system directly from perceptual input and are dominating in the field (Ramakrishnan, Al-Halah, and Grauman, 2020; Datta et al., 2020). According to the Habitat Challenge 2020 [3] PointGoal navigation benchmark, operating in the noisy environments state-of-the-art approaches can reach the destination in 71.7% of cases. But on the other hand, due to noisy environment and collisions, current agents still experience drifts over a long trajectories and could potentially benefit from robust visual odometry module.

1.2 Research Objective

In this master thesis, we outline the ideas to improve the efficiency and success rate of the navigation agents.

We aim to validate whether it is possible to achieve as good results in visual navigation under noisy conditions as they were achieved in environments without noise. For that, we formulate the set of research questions that we will investigate in experimental part of our work:

- incorporating meta-information available to the agent improves visual odometry model accuracy;
- removing redundant information by cropping out regions of visual observations that don't intersect may increase the visual odometry model robustnes;
- increasing the dataset size and diversity makes visual odometry model less sensitive to noise;
- visual odometry module can better generalize to noise distribution if separate model is used for estimating egomotion for particular action type.

1.3 Thesis Structure

The remainder of the thesis is structured as follows. In Chapter 2, we review the existing related work on PointGoal navigation. In Chapter 3, we present our approach, including the experiment setup and evaluation details. Our experiments results are discussed in Chapter 4. Finally, we make conclusive remarks in Chapter 5.

[3] https://aihabitat.org/challenge/2020

Chapter 2

Related Work

Autonomous navigation has been a subject of research in Robotics and Computer Vision for a long time (Moravec, 1984; Durrant-Whyte, Rye, and Nebot, 1996). With advances in Computer Vision and Deep Learning, there has been a renewed interest in the use of learning to derive navigation policies for a variety of tasks (such as rearrangement, visual navigation, and vision-and-language, and audio-visual navigation). PointGoal navigation (Chaplot et al., 2020; Ramakrishnan, Al-Halah, and Grauman, 2020; Datta et al., 2020), being one of the most fundamental among these task, has also been the subject of several prior works. We survey related works below.

2.1 Virtualization

Training and evaluating mobile robots in the real world environment is slow, resource intensive, hard to control and reproduce. For these reasons, such work has commonly been carried out in simulators (Savva et al., 2017; Xia et al., 2018; Savva et al., 2019). It allows researchers to tackle the Embodied AI tasks being equipped just with their own computer. Modern simulation environments provide a flexible APIs for experiment setup configuration and are optimized for speed. For example, the Habitat platform (Savva et al., 2019) - simulator that we used in our research, achieves several thousand frames per second (fps) running single-threaded, and can reach over 10,000 fps multi-process on a single GPU.

2.2 Classical vs Learned

Classical approaches decompose the problem into a sequence of sub-tasks, such as localization, mapping, planning, and control. Each of the sub-tasks is addressed separately and corresponding solutions are then composed into one pipeline. When properly tuned, such methods can perform well in cluttered environments, but are sensitive to noisy sensory input and fail to generalize to unseen environments.

Large and diverse photorealistic RGB-D datasets (Xia et al., 2018; Chang et al., 2017) together with simulators (Savva et al., 2017; Xia et al., 2018; Savva et al., 2019) that can run orders of magnitude faster than real time enable decades of agent experience to be collected in days. As a result, complementary to the classical approaches, the variety of methods that can learn the full navigation system directly from data appeared. Wijmans et al., 2020 showed that trained with sufficient amount of data and computational resources, learned approaches can significantly outperform their classical counterparts.

2.3 Visual Odometry

Our work is related to the prior work on localization and visual odometry (Kendall, Grimes, and Cipolla, 2015; Wang et al., 2017). Similar to our approach for visual odometry estimation is (Wang et al., 2017) that uses Recurrent Convolutional Neural Networks (RCNN) (Donahue et al., 2015) to directly estimate pose from raw RGB image sequences. In contrast, our visual odometry model estimates pose change between two consecutive observations and is used in conjunction with navigation policy. At every time step t the relative pose is computed as the aggregation of pose changes up to time step t.

2.4 Noiseless Setting [Solved]

Initial formulation of PointGoal navigation (as in Habitat Challenge 2019 [1]) assumes agents to have access to an egocentric RGB (or RGB-D) sensor and accurate oracle localization via a GPS+Compass sensor [2]. Moreover the action space is considered deterministic i.e. when the agent executes TURN_LEFT 30°, it turns exactly 30°, and MOVE_FORWARD 0.25m moves the agent exactly 0.25 meters forward. An episode is considered successful if the agent issues the STOP command within 0.2 meters of the goal.

Combination of recent advances in Deep Reinforcement Learning and distributed multi GPU parallelization set the state-of-the-art in above mentioned setting, essentially solving the task – near-perfect autonomous navigation in an unseen environment without access to a map, directly from an RGB-D camera and a GPS+Compass sensor, achieving 0.95 Success weighted by Path Length (SPL) 3.1 score with success rate 99.6% (Wijmans et al., 2020). Hence, so called noiseless setting of PointGoal navigation task is considered solved.

2.5 Noisy Setting

But no robot moves deterministically. Actuation error, surface properties such as friction, and a myriad of other sources of error introduce significant drift over a long trajectory. This poses a problem of PointGoal navigation under noisy actuation and without oracle localization (no GPS+Compass sensor). Nowadays the main emphasis is on increased realism and on *sim2real* predictivity (the ability to predict performance on a real robot from its performance in simulation) to close the gap between simulation and real world (Kadian et al., 2019).

Hence, in 2020's challenge benchmark the agent does not have a GPS+Compass sensor and must navigate solely using an egocentric RGB-D camera. This change elevates the need to perform RGB-D-based online localization.

Despite extensive research, navigation with noisy actuation and sensing continues to be an open frontier for research in Embodied AI.

State-of-the-art approaches in noiseless setting perform poorly under noisy conditions. Therefore the new approaches designed to cope with actuation and sensing noise appeared and set the highest scores in PointGoal navigation under noisy actuation and sensing.

[1]https://aihabitat.org/challenge/2019

[2]Habitat simulator sensor that provides access to compass and GPS data.

The Neural SLAM by Chaplot et al., 2020 is the first approach that integrated learning into classical modular SLAM components. More specifically, the architecture comprises of a learned Neural SLAM module, a global policy, and a local policy. The learned Neural SLAM module uses RGB observations and motion sensor readings to produce free-space maps and agent pose estimates. On top of this, the global policy produces long-term goals. The long-term goals are then consumed by the geometric path-planner to generate short-term goals for the local policy. In the end, the local policy maps RGB observations to actions that the agent should execute to reach the short-term goal. Designed for exploration it has shown state-of-the-art performance when transferred to the PointGoal task in Habitat Challenge 2019 benchmark. But in contrast to our problem setting, the Neural SLAM relies on motion sensor which estimates the robot pose as the agent moves.

Built on top of Neural SLAM architecture, Occupancy Anticipation by Ramakrishnan, Al-Halah, and Grauman, 2020 uses its egocentric RGB-D observations to infer the occupancy state beyond the visible regions. Method leverages context in both the egocentric views and top-down maps as well as a pose estimator trained to predict the change in agents location and orientation under noisy conditions. In doing so, the agent builds its spatial awareness more rapidly, which facilitates efficient exploration and navigation in 3D environments.

Integrating Egocentric Localization by Datta et al., 2020 conceptually divides learning agent dynamics or odometry from task-specific navigation policy. Developed model had an internal visual odometry module that estimates changes in position and heading from consecutive visual observations. This allows the agent to maintain a noisy, but up-to-date estimate of pose by integrating the per-action ego-motion estimates along its trajectory. Proposed method showed that it's possible to effectively navigate in previously unseen environments without explicitly building a map. It was shown that being equipped with ego-localization module, agent is able to make progress towards the goal by a degree that reasonably matches that of the "oracle" agent (SoftSPL=0.813 vs SoftSPL=0.865). Moreover, with access to ground-truth localization agent reaches the goal in 94.8% of episodes achieving 0.866 SPL score compared to agent with ego-localization module scoring 53.5% Success and 0.508 SPL. Meaning that accurate visual odometry estimation is crucial for agent to navigate successfully. Complementary to previous research we focus on designing visual odometry model robust to noise and collisions to alleviate trajectory drifts caused by incorrect pose estimation.

Chapter 3

Approach

We further investigate the idea suggested by Datta et al., 2020 to use the visual odometry module as a drop-in replacement of ground-truth localization sensor. Our pipeline consists of two components: navigation policy that given observations at time step t decides which action to take to reach the goal and visual odometry module that estimates the change in location and orientation between two consecutive observations. We train policy and odometry separately and then replace the ground-truth localization signal with odometry module estimates without further fine-tuning.

Detailed experiment setup, model architectures along with training and evaluation details are described in the subsequent sections.

3.1 Experiment Setup

To simulate navigation experiments we use the Habitat platform (Savva et al., 2019). The results of Habitat Challenge 2020 indicate that the benchmark is far from being solved. Thus, the PointGoal navigation task specifications for Habitat Challenge 2021 remained unchanged except for the agent's camera's tilt angle. Our experimental setup follows the Habitat Challenge 2021 benchmark. An agent is spawned at a random starting position and orientation in an unseen environment and asked to navigate to target coordinates specified relative to the agent's start location. Target coordinates are specified once at the start of the episode and doesn't update during the navigation. No ground-truth map and GPS+Compass sensor is available, the agent must only use its visual sensory input (an RGB-D camera) to navigate.

Driven by experiments in reality the agent's specification matches
the LoCoBot's [1] specification. Base radius is 0.18m and height is 0.88m.

[1]LoCoBot is a low-cost mobile manipulator suitable for both navigation and manipulation (http://www.LoCoBot.org/).

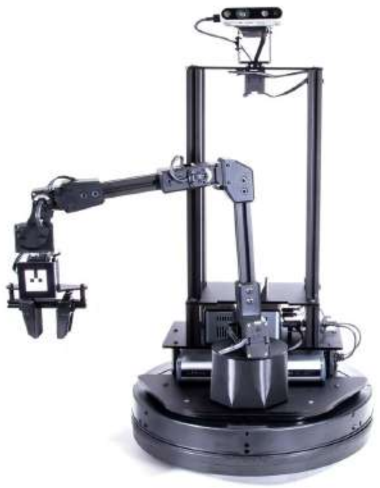


FIGURE 3.1: LoCoBot (an open source low cost robot).

The action space consists of four actions:

- STOP - finishes navigation episode;
- MOVE_FORWARD - moves the agent $(0.25 + \epsilon_1)$ meters forward;
- TURN_LEFT - turns the agent $(30 + \epsilon_2)$ degrees left;
- TURN_RIGHT - turns the agent $(30 + \epsilon_3)$ degrees right.

were $\epsilon_1, \epsilon_2, \epsilon_3$ are the noise values, acquired by benchmarking the LoCoBot robot. Even though there are four discrete actions it doesn't guarantee the deterministic position and orientation transformation. The agent may experience rotation and translation noise both while rotating and moving forward. As a result, identical action sequences can lead to vastly different final locations.

The RGB and Depth sensor readings are noisy as well. Camera's resolution is 360×640 pixels with 70 degrees horizontal field of view and -20 degrees tilt angle. Depth sensing is clipped to [0.1m, 10m]. Sliding along the walls is disabled.

3.2 Model Architecture

3.2.1 Navigation Policy

Two layer Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) model, with a ResNet-18 (He et al., 2016) encoder is used to approximate the navigation policy. Policy observation space consists of Depth sensor and idealistic GPS+Compass sensor (ground-truth localization, that is replaced by the visual odometry estimates during evaluation). Before passing through the feature encoder, visual observations are transformed using ResizeShortestEdge and CenterCropper observation transforms. Where first resizes the shortest edge of the input to 256 pixels while maintaining aspect ratio and second center crops the input to 256×256 pixels.

[1]LoCoBot image source: https://www.generationrobots.com/en/403494-locobot-autonomous-mobile-manipulator-ros-compatible.html.

We leverage Decentralized Distributed Proximal Policy Optimization (DD-PPO) algorithm to train the policy, as the large scale training allows to find an optimal navigation policy faster. We consider navigation policy training setup introduced by Wijmans et al., 2020 as the initial setup in our experiments. We train agents for more than 2 billion steps of experience with 128 workers.

3.2.2 Visual Odometry

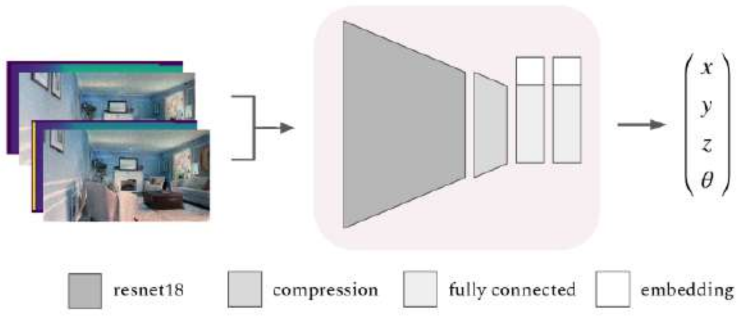


FIGURE 3.2: Visual odometry module.

The visual odometry model is represented as ResNet-18 encoder followed by a compression block with two fully connected layers on top (size 512), taking the pair of RGB-D frames as input and predicting the relative pose change (see Figure 3.2). Compression block consists of 3×3 convolutional kernel followed by GroupNorm (Wu and He, 2018) layer and ReLU activation function. We also replaced BatchNorm (Ioffe and Szegedy, 2015) layers in the canonical ResNet-18 architecture with GroupNorm and added a DropOut (Srivastava et al., 2014) with 0.2 probability between fully connected layers. Meta-information about action and/or collision is incorporated as an embedding - vector of fixed length that is concatenated to the flattened output that comes from the compression layer. The DropOut is not applied to neurons that correspond to embedding vector. We also downscale the observations twice from 360 × 640 pixels to 180 × 320 pixels during training and evaluation.

The model is trained with batch size 16, Adam optimizer with learning rate 0.0001 and MSE Loss 3.3 for translation and rotation. To speed up training we run experiments in distributed mode with 2 parallel workers on the node with two NVidia GeForce RTX 3090.

3.2.3 Navigation Policy and Visual Odometry Integration

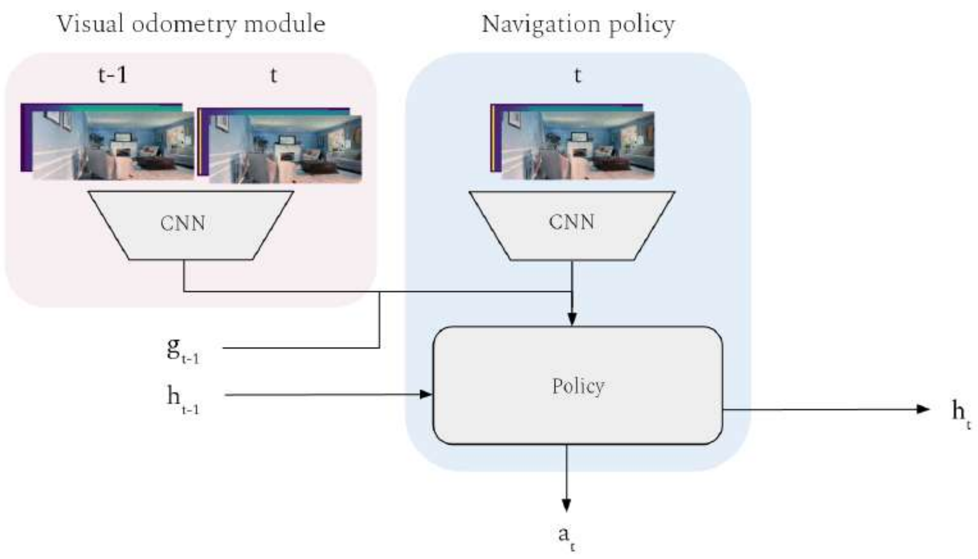


FIGURE 3.3: PointGoal navigation agent architecture.

Figure 3.3 shows how the RL navigation policy and visual odometry module are integrated together. Where g_{t-1} - goal coordinates (wrt. current pose), a_t - action to take, h_{t-1} and h_t - hidden states. Predicted pose change is subtracted from the goal location and the result is feed into policy as a new goal location. The initial goal location estimate is equal to ground truth goal location.

3.3 Datasets

3.3.1 Navigation Policy

FIGURE 3.4: Gibson scenes.

We use the Habitat Challenge subset of Gibson [2] dataset for training the RL navigation policy. It consists of meshes that do not exhibit significant reconstruction artifacts such as holes or texture quality issues (with ratings of 4 or higher). The

[2]Gibson scenes image source: http://gibsonenv.stanford.edu/database.

scenes were collected from real indoor spaces using 3D scanning and reconstruction. There are 86 scenes in total: 72 in train split and 14 in validation split. Subset of Gibson scenes is visualized in Figure 3.4.

For PointGoal task we use Gibson navigation episodes corresponding to Sim2LoCoBot experiment configuration pointnav_gibson_v2.zip [3] (meets LoCoBot specification).

3.3.2 Visual Odometry

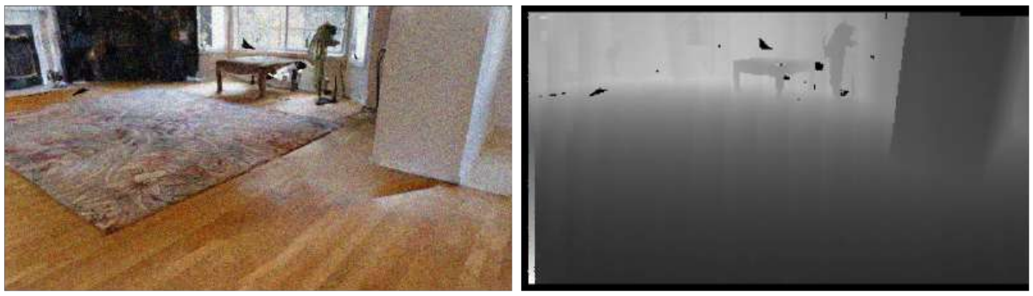


FIGURE 3.5: RGB-D observation.

Visual odometry model is trained and evaluated on generated statical dataset that consists of pairs of observations navigation agent receives before and after taking a particular action. We employ the visual odometry dataset collection protocol suggested by Datta et al., 2020, where the agent with access to ground-truth localization is used to unroll trajectories from which the pairs of RGB-D frames with meta information about collisions, actions taken and egomotions are uniformly sampled. But we use shortest path follower agent to unroll trajectories instead of DD-PPO agent used by (Datta et al., 2020). Also, our dataset generation code supports multi-core parallelization that significantly speed-ups the process. In parallel mode each worker receives the scene and samples the frames from corresponding navigation episodes. An example of RGB-D observation is visualized in Figure 3.5.

Training dataset was collected by sampling 20% of pairs of observations from 11000 navigation episodes (130881 pairs of observations in total). Validation dataset was collected by sampling 75% of pairs of observations from 994 navigation episodes (34339 pairs of observations in total). Figure 3.6 shows the distributions of actions in training and validation datasets respectively.

To study the impact of large scale training we increased the training dataset size from 130881 pairs of observations to 641052 keeping the actions and collisions distributions the same.

3.4 Metrics and Evaluation

3.4.1 Navigation Policy

After calling the STOP action, the agent's navigation performance is evaluated using the Success weighted by Path Length metric

$$SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i, l_i)} \tag{3.1}$$

[3]https://dl.fbaipublicfiles.com/habitat/data/datasets/pointnav/gibson/v2/pointnav_gibson_v2.zip

and it's soft version SoftSPL

$$SoftSPL = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{d_{T_i}}{d_{init_i}}\right)\left(\frac{l_i}{max(p_i, l_i)}\right) \tag{3.2}$$

where, l_i = length of shortest path between goal and target for an episode i, p_i = length of path taken by agent in episode i, S_i = binary indicator of success in episode i, d_{init_i} and d_{T_i} denote the geodesic distance to target upon episode start and termination. An episode is considered successful if on calling the STOP action, the agent is within $0.36m$ (2 x agent-radius) of the goal position.

3.4.2 Visual Odometry

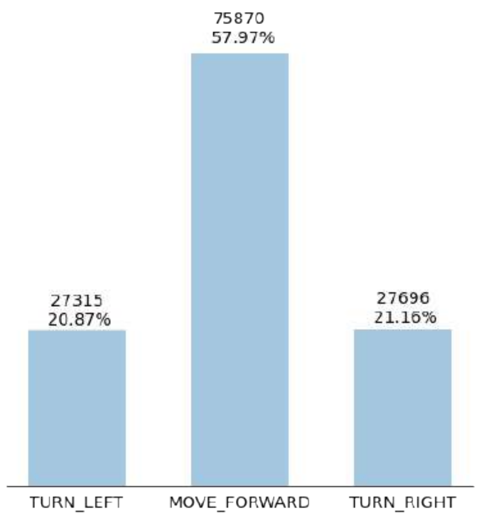
For the visual odometry we consider the regression loss i.e. how good the predicted pose change matches true pose.

$$L = \frac{1}{N}\sum_{i=1}^{N}\left((x-\hat{x})^2 + (y-\hat{y})^2 + (z-\hat{z})^2\right) + \frac{1}{N}\sum_{i=1}^{N}(\theta-\hat{\theta})^2 \tag{3.3}$$

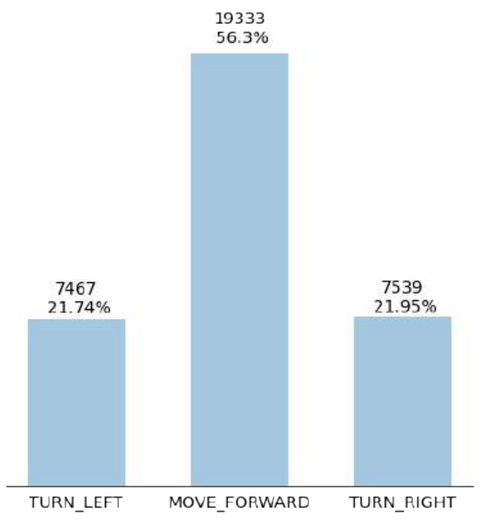
where x, y, z represent the xyz coordinate of the agent measured in metres and θ represents the orientation of the agent in radians; (x, y, z, θ) is the ground truth agent pose, $(\hat{x}, \hat{y}, \hat{z}, \hat{\theta})$ is the estimated pose, N number of RGB-D pairs in the dataset.

We use Mean Absolute Error (MAE) for measuring model accuracy and report total MAE for all actions in total and for each type of action separately.

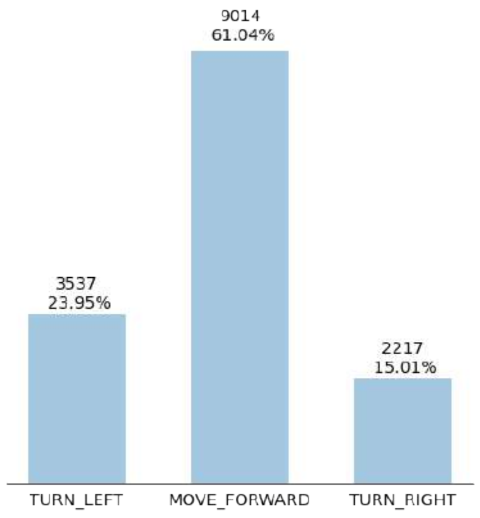
$$MAE = \frac{1}{N}\sum_{i=1}^{N}(|x-\hat{x}| + |y-\hat{y}| + |z-\hat{z}|) + \frac{1}{N}\sum_{i=1}^{N}|\theta-\hat{\theta}| \tag{3.4}$$



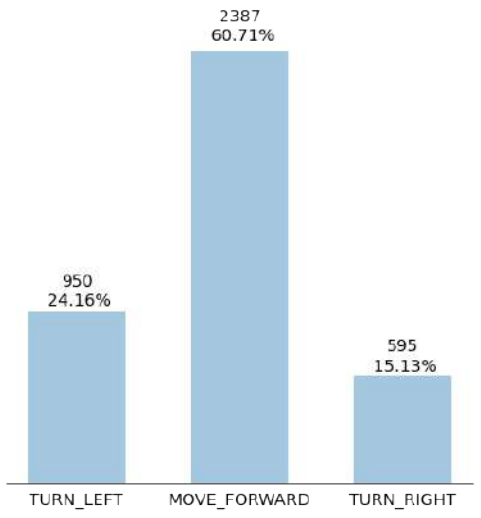
(A) Training dataset.



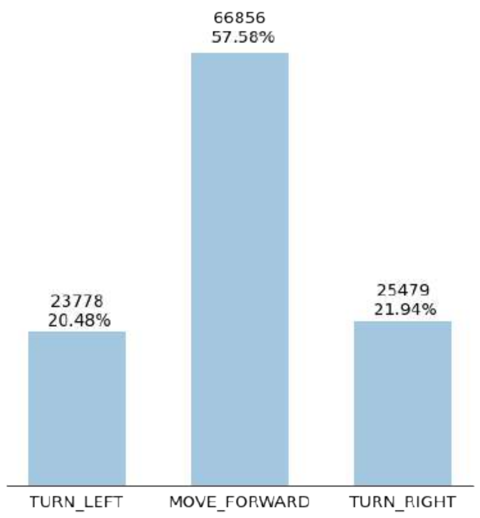
(B) Validation dataset.



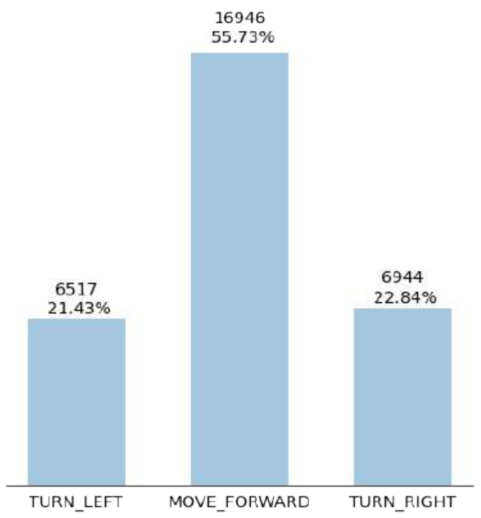
(C) Training dataset (collisions).



(D) Validation dataset (collisions).



(E) Training dataset (no collisions).



(F) Validation dataset (no collisions).

FIGURE 3.6: Actions distribution in visual odometry training and validation datasets.

Chapter 4

Experiments

In this chapter we present our experimental results on designing robust visual odometry module for realistic PointGoal navigation.

In Section 4.1 we first study the importance of incorporating mata-information (potentially) available to the agent by adding action and collision embeddings. Secondly, we check if the model robustness can be improved by cropping out parts of the visual observations that don't intersect. Then we analyse the impact of enriching the dataset by reflecting the observations along X axis and swapping source and target frames for rotations. Also we train a separate model for each type of actions. Finally, we show how the model generalization performance changes at large scale training.

In Section 4.2 we analyse navigation performanve of policy with standard and narrowed success region radius and report the navigation metrics for visual odometry models from Section 4.1.

In Section 4.2.2 we report our results in Habitat Challenge 2021 benchmarks.

4.1 Visual Odometry

We run our experiments using visual odometry model architecture described in Section 3.2. We start from the baseline model that consists of a ResNet-18 encoder followed by a compression block with two fully connected layers on top and then describe the impact of proposed regularization techniques in corresponding subsections. All experiments were run for 50 epochs. We compare the model performance using MAE 3.4 between ground-truth and estimated egomotion for all types of actions in total and for each type of actions separately. We plot the metrics chart for all experiments in the subsection and report validation metrics values on 20 and 40 epochs. Per-action metric charts are reported in the Appendix A.2. We add Table 4.7 with aggregated metrics values across all experiments from Section 4.1 in the end of the section. Corresponding agent's navigation metrics are reported in Table 4.8 of Section 4.2.

4.1.1 Embeddings

In this section we study the impact of adding meta-information during training in the form of action and collision embeddings. Embeddings are represented as a fixed vectors of lenght 8 that are concatenated to the flattened output from the feature encoder. We analyze two possible ways of incorporating meta-information: concatenating the embedding to the first fully connected layer that goes after encoder and concatenating the embedding to all fully connected layers.

Following experiments were launched:

- baseline with action embedding (bs + act_emb);
- baseline with collision embedding (bs + col_emb);
- baseline with action and collision embeddings (bs + col_emb + act_emb);
- baseline with action embedding concatenated to every fully collected layer (bs + act_emb 2fc);
- baseline with action and collision embeddings concatenated to every fully collected layer (bs + col_emb + act_emb 2fc).

TABLE 4.1: Embeddings: visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs	20	4.61	5.41	3.48	3.67	2.37	1.58	3.61	3.17
	40	4.10	4.53	3.47	3.65	1.85	1.29	2.65	2.49
bs + col_emb	20	5.51	6.99	3.48	3.73	2.98	2.15	3.90	4.18
	40	4.65	5.45	3.56	3.67	2.40	1.64	3.28	3.45
bs + act_emb	20	3.13	2.86	3.45	3.48	1.32	0.99	1.74	1.76
	40	2.89	2.39	3.43	3.65	1.15	0.78	1.62	1.65
bs + col_emb + act_emb	20	3.10	2.80	3.37	3.62	1.37	1.02	1.79	1.87
	40	3.00	2.56	3.48	3.65	1.26	0.84	1.74	1.85
bs + act_emb 2fc	20	3.00	2.67	3.39	3.47	1.25	0.96	1.58	1.68
	40	2.89	2.43	3.41	3.58	1.16	0.82	1.59	1.63
bs + col_emb + act_emb 2fc	20	3.08	2.80	3.38	3.49	1.24	0.94	1.59	1.65
	40	2.87	2.43	3.31	3.57	1.17	0.85	1.55	1.60

The results in Table 4.1 and Figure 4.1 and Figure 4.2 show that adding meta-information in form of action and collision embeddings helps neural network to train faster. We believe that it serves as a switch and activates the neurons that correspond to a particular action type. The training and validation metrics curves are much lower than the beseline. And the way how the meta-information is incorporated also matters. We see that, for example, model with action embedding concatenated to every fully collected layer (bs + act_emb 2fc) has lower MAE than model with action embedding concatenated only to the first fully collected layer (bs + act_emb).

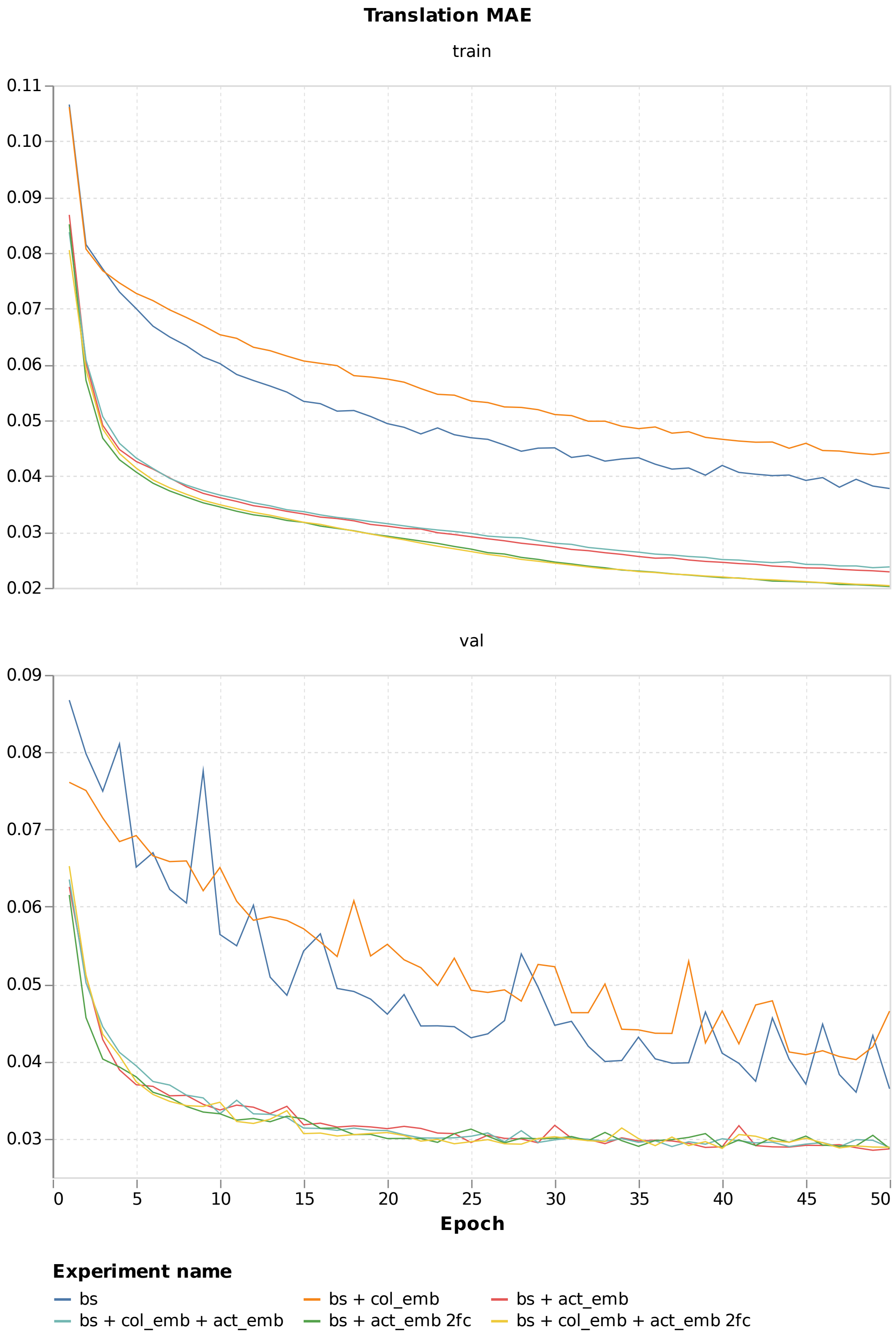


FIGURE 4.1: Embeddings: translation MAE.

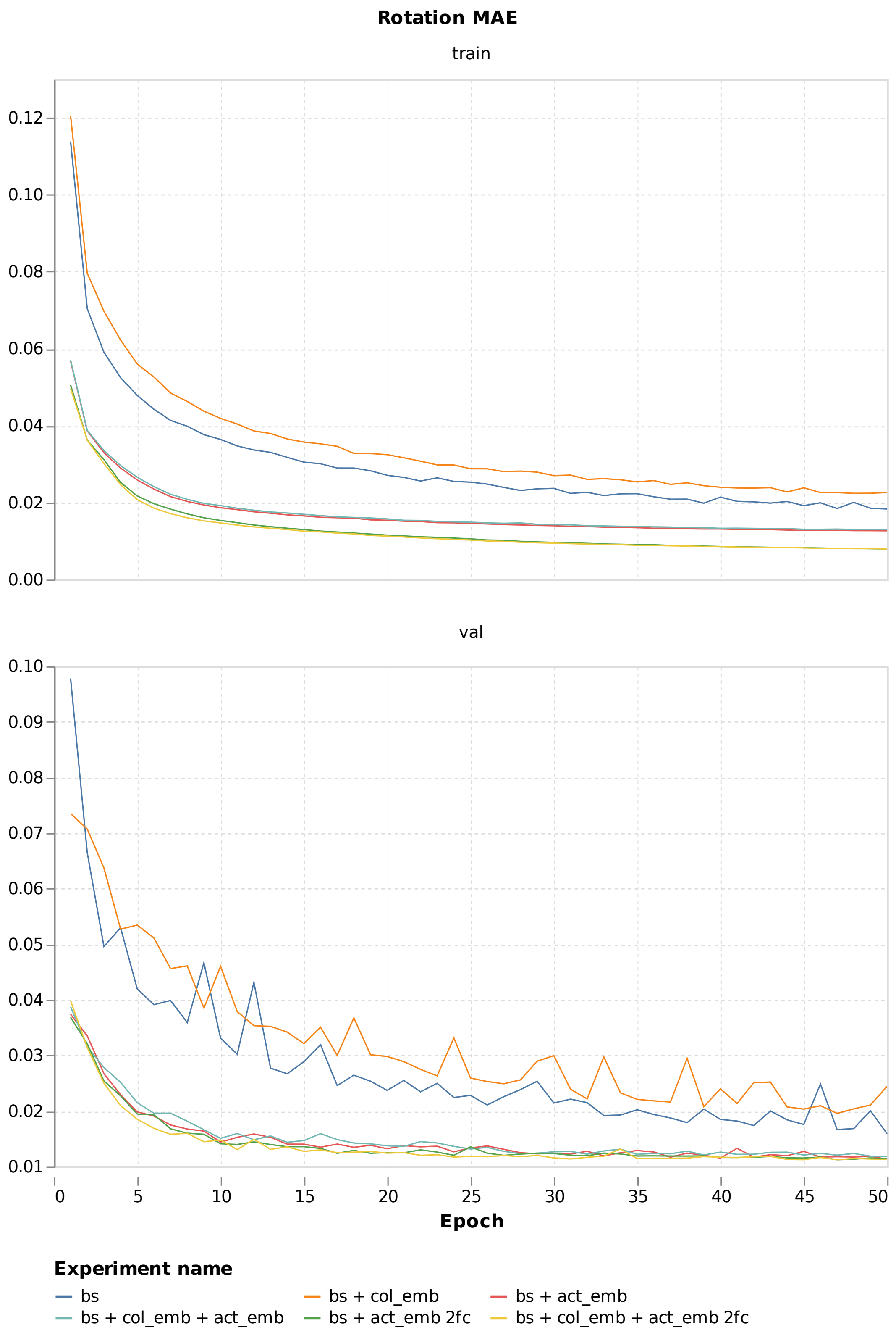


FIGURE 4.2: Embeddings: rotation MAE.

4.1.2 Visual Observations Intersection Region Crop

In this section we raise the question whether the whole region of visual observation is important for egomotion estimation. To answer the question we run experiments where we use only intersection region between visual observations for training. The hypothesis is that by cropping only an intersection region we may remove redundant information and thus achieve more accurate results. We use the model (bs + act_emb 2fc) from previous experiments as a baseline.

We crop observations as follows:

- for MOVE_FORWARD action center crop;
- for TURN_LEFT action crop left part of the source frame and right part of the target frame;
- for TURN_RIGHT action crop right part of the source frame and left part of the target frame.

Figure 4.3 demonstrates the example of 320 $\times$ 450 observations crop. Figure 4.4 demonstrates the example of 320 $\times$ 350 observations crop. The examples of visual observation crop for other types of actions are added in Appendix A.1.

Following experiments were launched:

- baseline with observations crop 320$\times$450 downsampled to 160$\times$225 (bs + act_emb 2fc + 320x450->160x225);
- baseline with observations crop 320$\times$450 downsampled to 180$\times$320 (bs + act_emb 2fc + 320x450->180x320);
- baseline with observations crop 320$\times$320 downsampled to 180$\times$320 (bs + act_emb 2fc + 320x350->180x320).

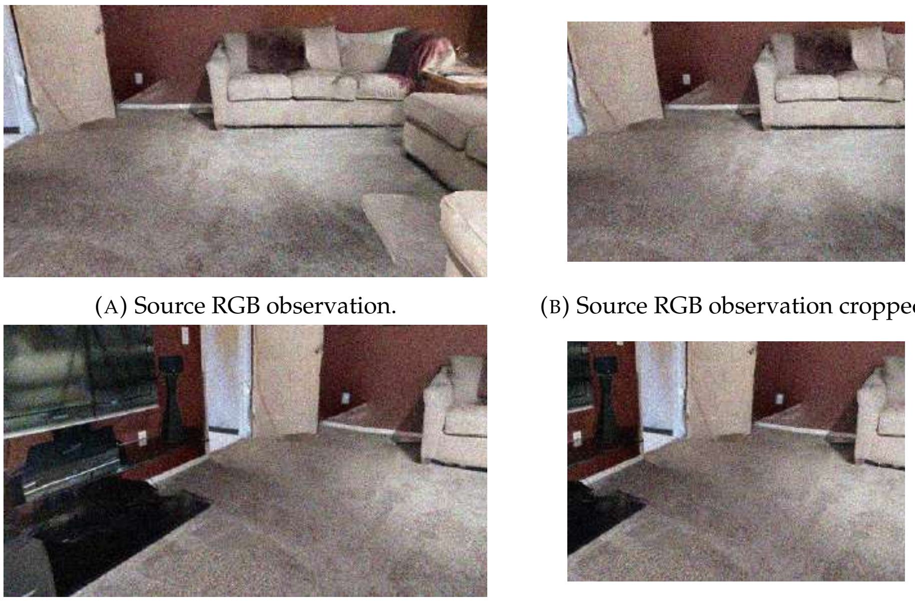
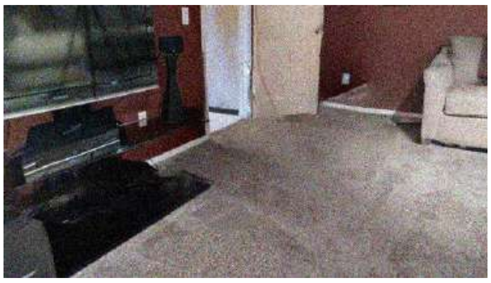

(A) Source RGB observation. (B) Source RGB observation cropped.
(C) Target RGB observation. (D) Target RGB observation cropped.

FIGURE 4.3: TURN_LEFT crop 320 $\times$ 450.

(A) Source RGB observation.

(B) Source RGB observation cropped.


(C) Target RGB observation.

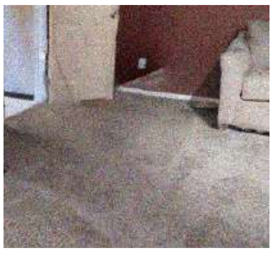

(D) Target RGB observation cropped.

FIGURE 4.4: TURN_LEFT crop 320 × 350.

TABLE 4.2: Crop: visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs + act_emb 2fc	20	3.00	2.67	3.39	3.47	1.25	0.96	1.58	1.68
	40	2.89	2.43	3.41	3.58	1.16	0.82	1.59	1.63
bs + act_emb 2fc + 320x450->160x225	20	3.28	3.41	3.03	3.19	1.24	1.09	1.40	1.49
	40	3.14	3.18	3.01	3.16	1.15	0.95	1.38	1.44
bs + act_emb 2fc + 320x450->180x320	20	3.12	3.15	3.02	3.16	1.26	1.02	1.57	1.57
	40	2.94	2.89	3.00	3.00	1.18	1.00	1.41	1.42
bs + act_emb 2fc + 320x350->180x320	20	3.17	3.58	2.59	2.67	1.18	1.14	1.19	1.27
	40	2.76	3.05	2.36	2.41	0.98	0.95	1.02	1.00

Models that are trained on the whole images usually learn to estimate egomotion for MOVE_FORWARD action better than for rotations (high color contrast in Table 4.2 between Forward and Left, Right columns both for rotations and translations). In our series of experiments we were incrementally removing regions of the image that don't intersect increasing the share of the intersection region between source and target frames. In (bs + act_emb 2fc + 320x450->180x320) experiment we crop the region of 320 × 450 pixels and downscale it to 180 × 320 pixels, in (bs + act_emb 2fc + 320x350->180x320) experiment we crop the region of 320 × 350 pixels and downscale it to 180 × 320 pixels. Experiments results show that model trained on the observation crops that have high share of intersection can estimates the egomotion for rotations better than the model trained on the whole images, but on the other hand the estimation performance for MOVE_FORWARD action degradates. We may also see that downscaling to 180 × 320 pixels works better than downscaling to 160 × 225 pixels. The difference in the performance can be better spot in the plots in Appendix A.2.2.

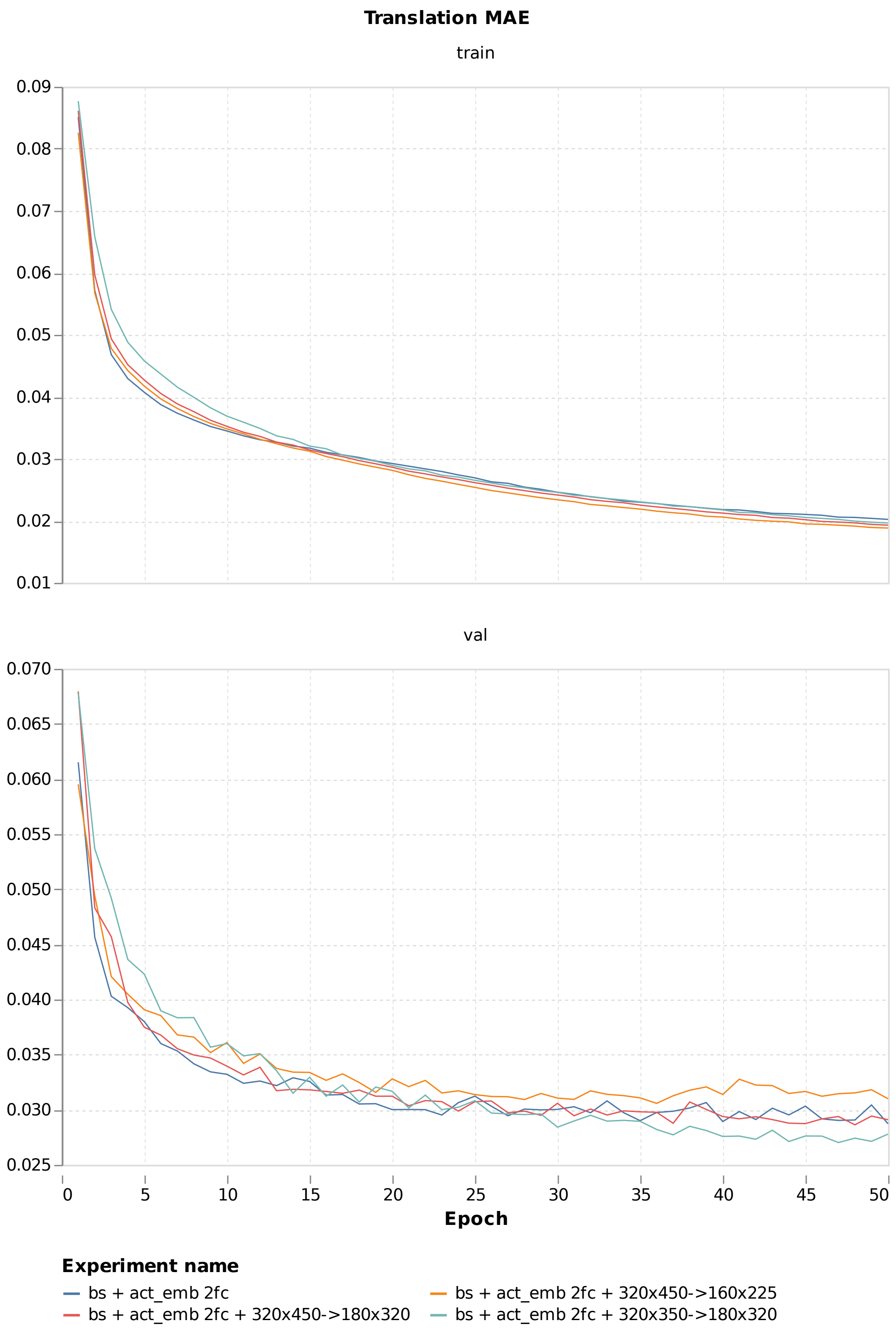


FIGURE 4.5: Crop: translation MAE.

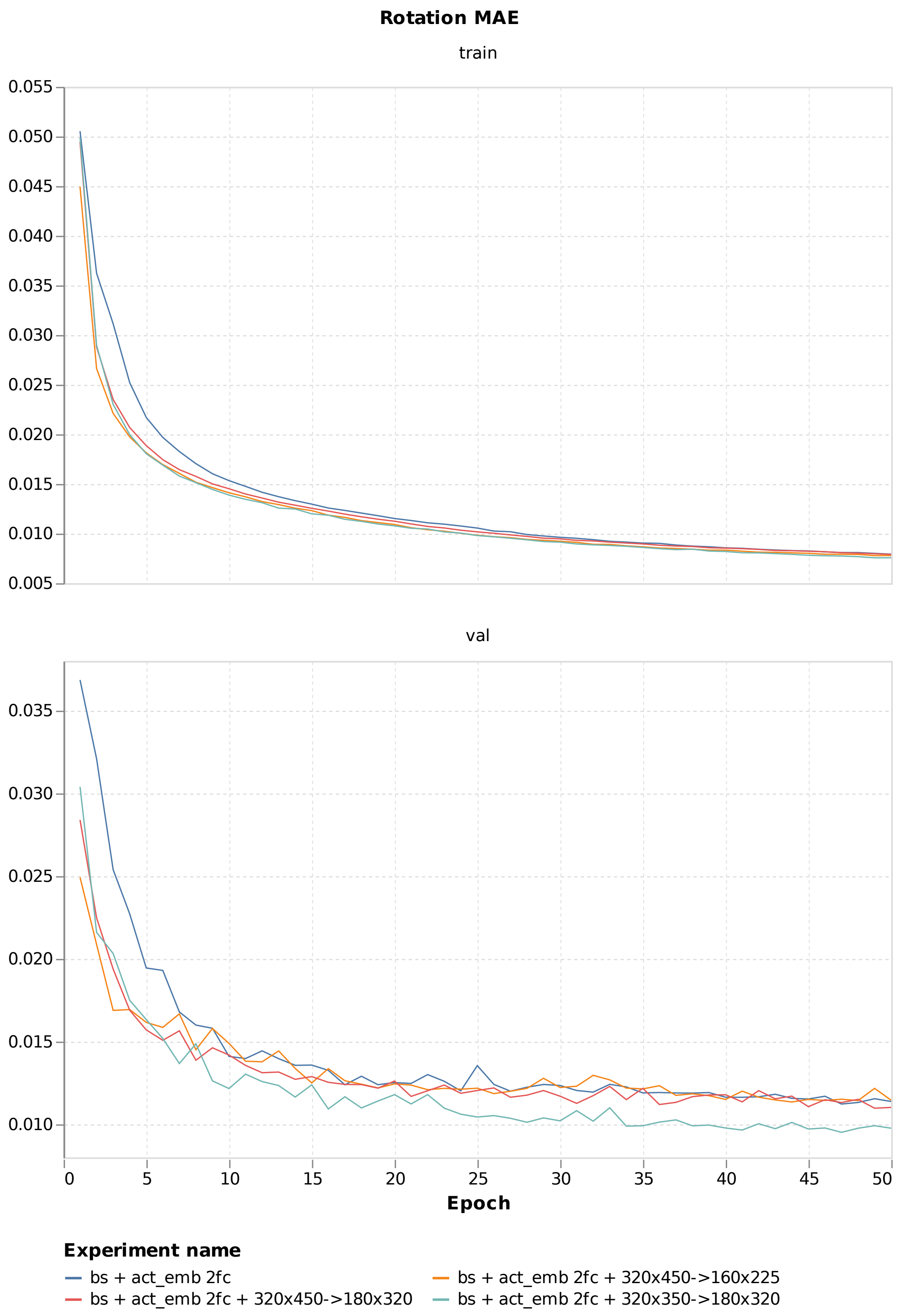


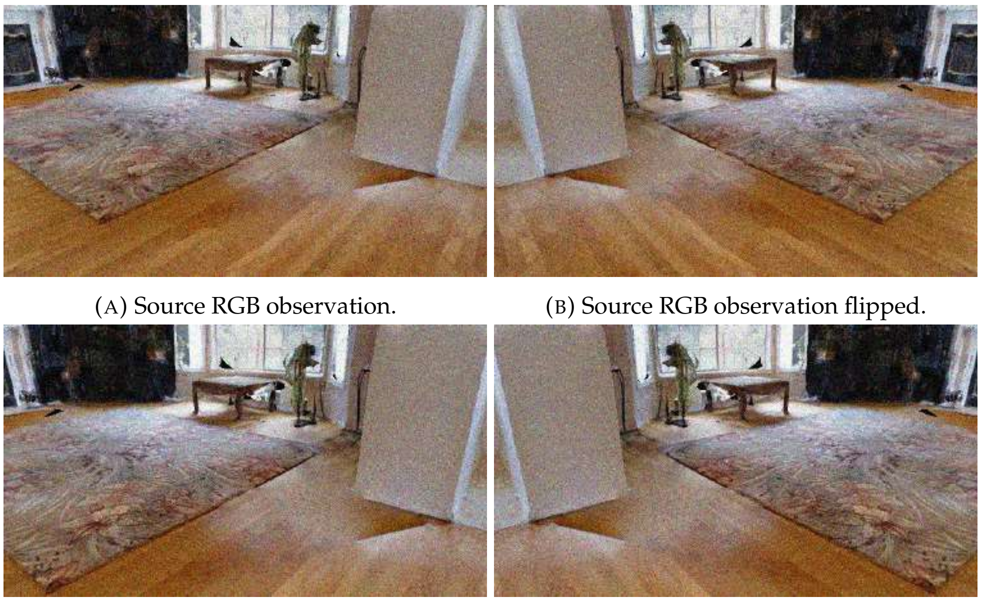
FIGURE 4.6: Crop: rotation MAE.

4.1.3 Frames Flip

In this section we analyze the effect of enriching the training dataset diversity by randomly reflecting 50% of visual observations along the X axis during training as demonstrated in Figure 4.7. The same type of augmentation can also be applied at test-time and compute the final egomotion as the average egomotion between original and flipped frames reflected back. We use the model (bs + act_emb 2fc) from Subsection 4.1.1 as a baseline.

Following experiments were launched:

- baseline with vertical flip augmetarion (bs + act_emb 2fc + vflip).



(A) Source RGB observation. (B) Source RGB observation flipped.

(C) Target RGB observation. (D) Target RGB observation flipped.

FIGURE 4.7: MOVE_FORWARD flip.

TABLE 4.3: Flip: visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs + act_emb 2fc	20	3.00	2.67	3.39	3.47	1.25	0.96	1.58	1.68
	40	2.89	2.43	3.41	3.58	1.16	0.82	1.59	1.63
bs + act_emb 2fc + vflip	20	2.92	2.54	3.37	3.44	1.14	0.85	1.51	1.53
	40	2.73	2.30	3.23	3.32	1.03	0.73	1.41	1.41

Such type of augmentation improves both translation and rotation MAE for all checkpoints. Thus, we may conclude that using the diverse training dataset results in more accurate visual odometry model.

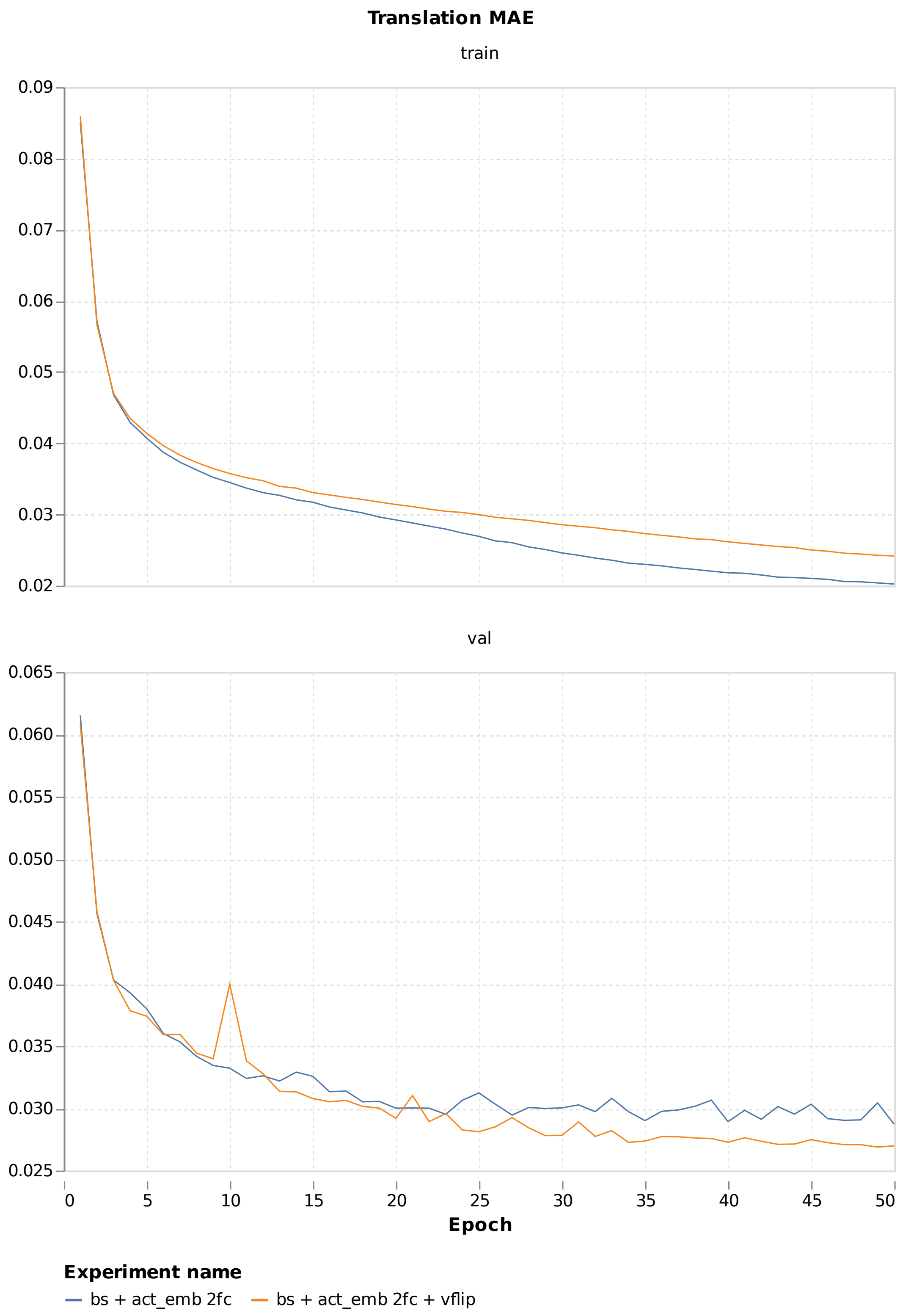


FIGURE 4.8: Flip: translation MAE.

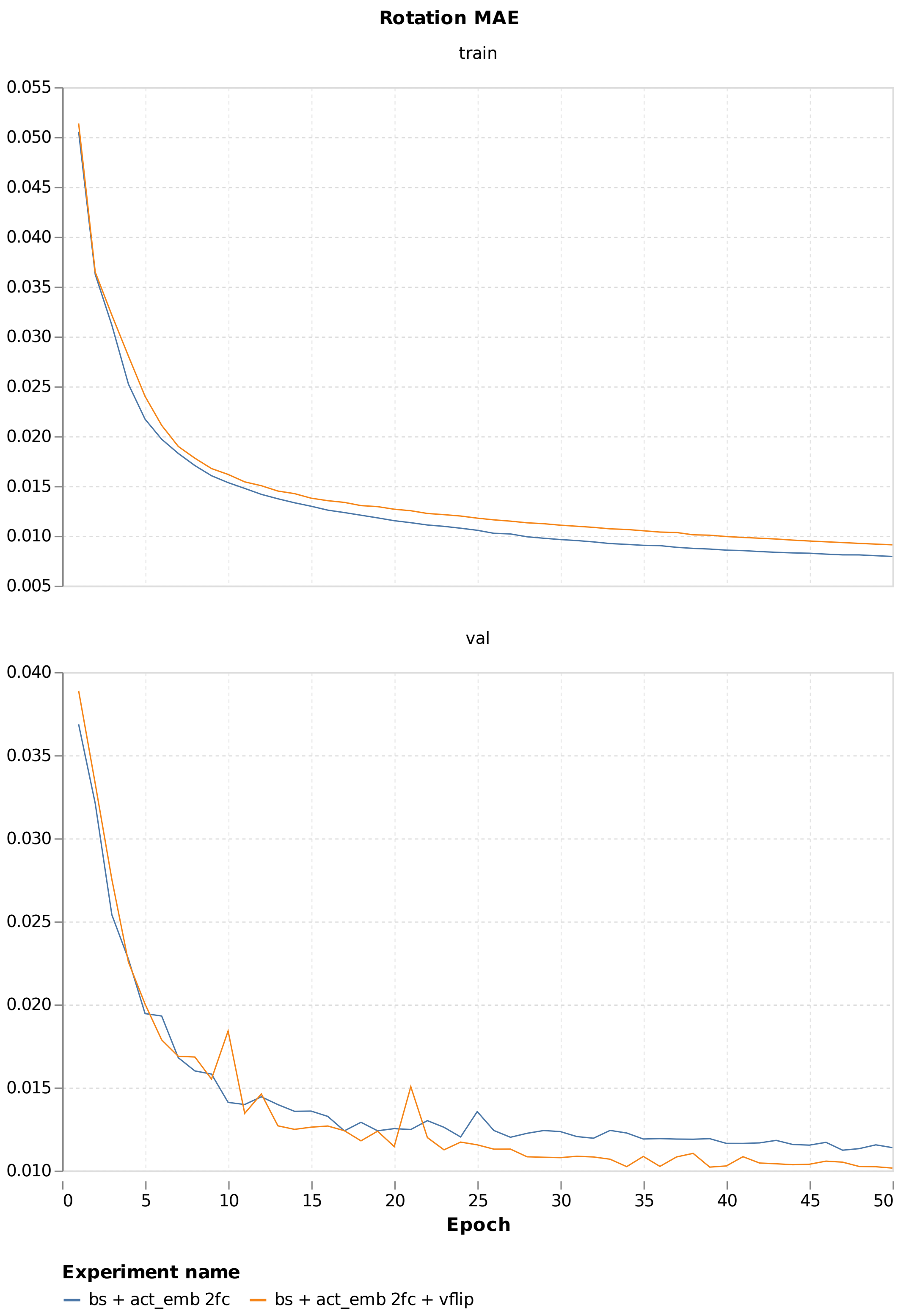


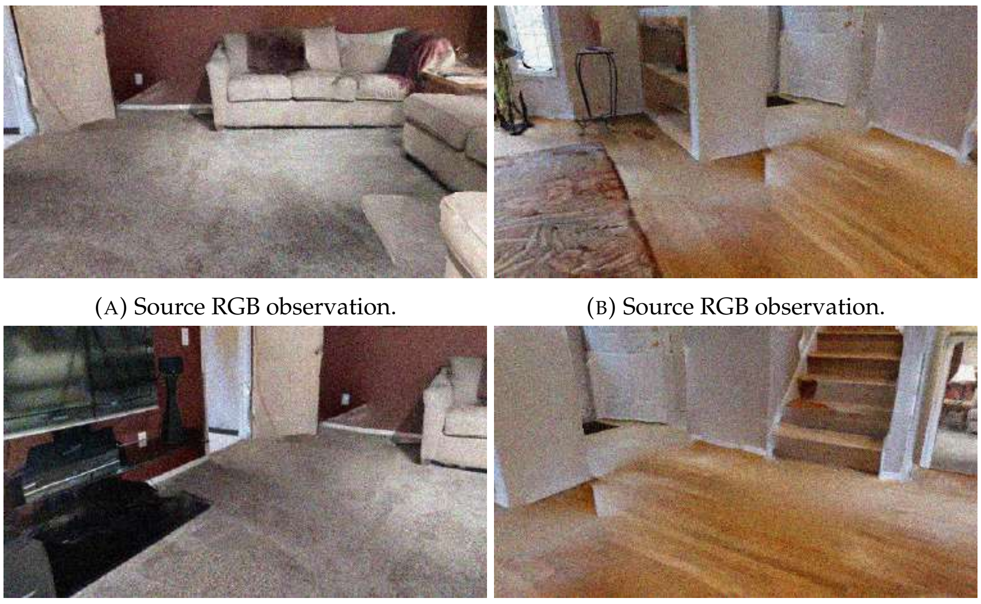
FIGURE 4.9: Flip: rotation MAE.

4.1.4 Frames Swap

In this section we analyze the effect of enriching the training dataset diversity by adding inverse observations for rotations during training as demonstrated in Figure 4.10. Specifically, we swap the source and target frames for TURN_LEFT and TURN_RIGHT observation pairs thus increasing the dataset size and changing the actions distribution as shown in Figure 4.11. The same type of augmentation can also be applied at test-time and compute the final egomotion as the average egomotion between original and swapped frames. We use the model (bs + act_emb 2fc + vflip) from Subsection 4.1.3 as a baseline.

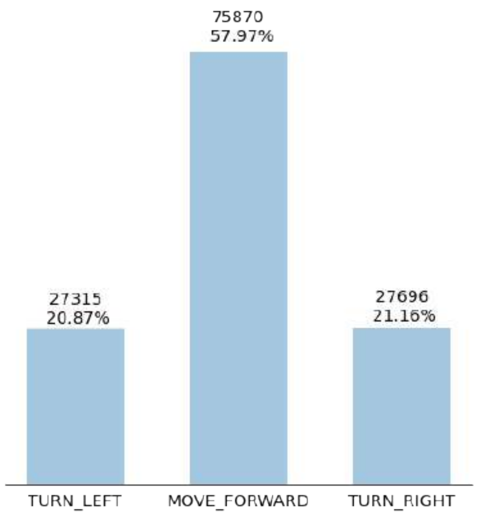
Following experiments were launched:

- baseline with fames swap during training (bs + act_emb 2fc + vflip + inv_rot).

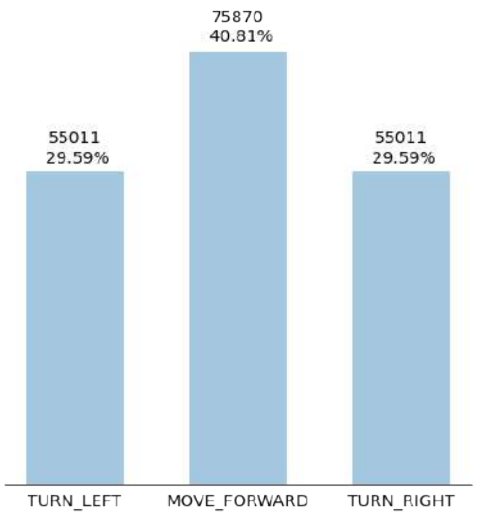

(A) Source RGB observation. (B) Source RGB observation.

(C) Target RGB observation. (D) Target RGB observation.

FIGURE 4.10: TURN_LEFT - left column, TURN_RIGHT - right column.



(A) Actions distribution in the original training dataset.



(B) Actions distribution in the training dataset with inverse rotations added.

FIGURE 4.11: Visual odometry datasets action distribution.

TABLE 4.4: Swap: visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs + act_emb 2fc + vflip	20	2.92	2.54	3.37	3.44	1.14	0.85	1.51	1.53
	40	2.73	2.30	3.23	3.32	1.03	0.73	1.41	1.41
bs + act_emb 2fc + vflip + inv_rot	20	2.89	2.62	3.20	3.30	1.06	0.82	1.37	1.37
	40	2.76	2.42	3.17	3.23	0.96	0.69	1.28	1.32

Enreaching the training dataset with inversed source and target frames makes dataset less imbalanced and improves rotation MAE.

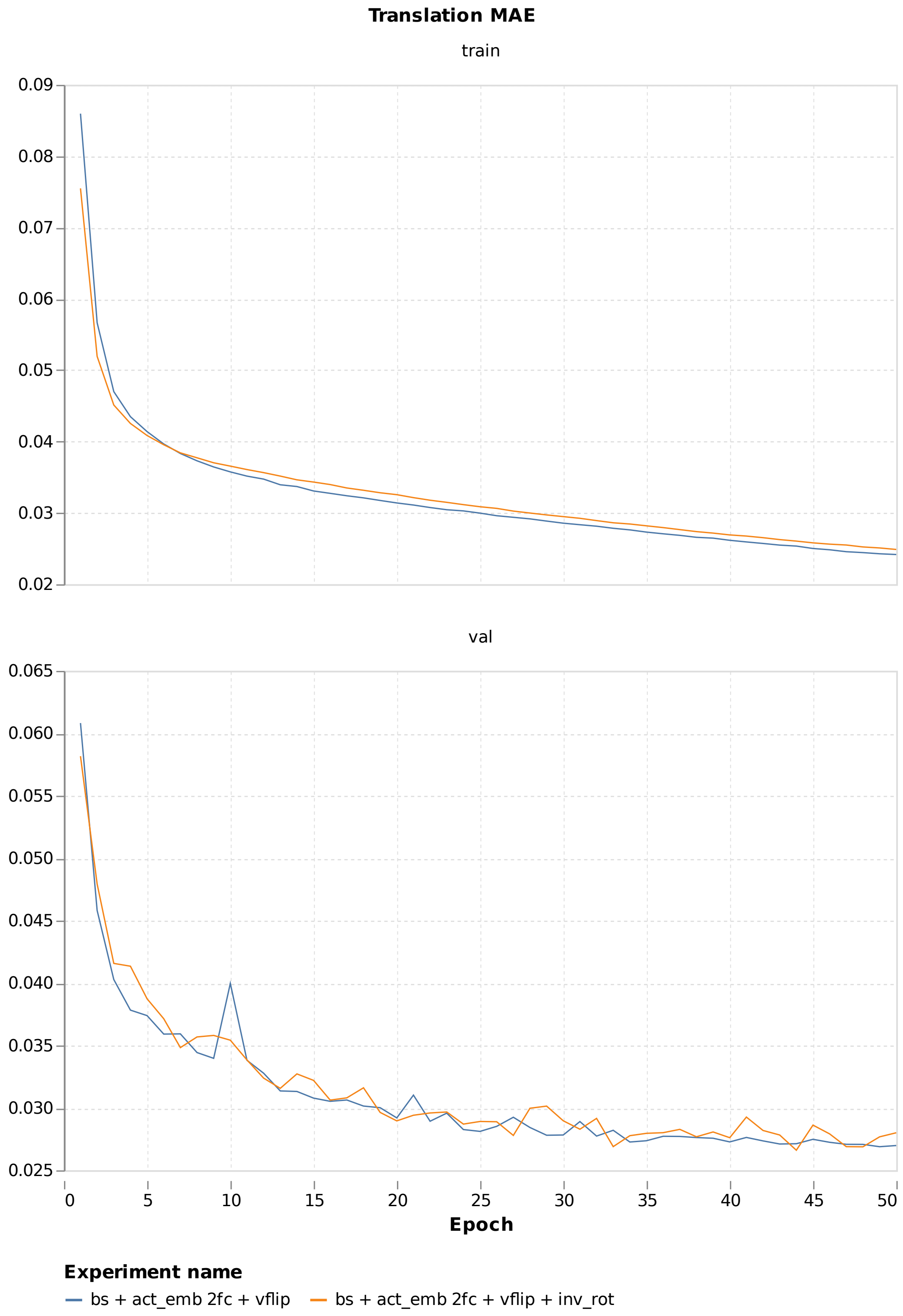


FIGURE 4.12: Frames swap: translation MAE.

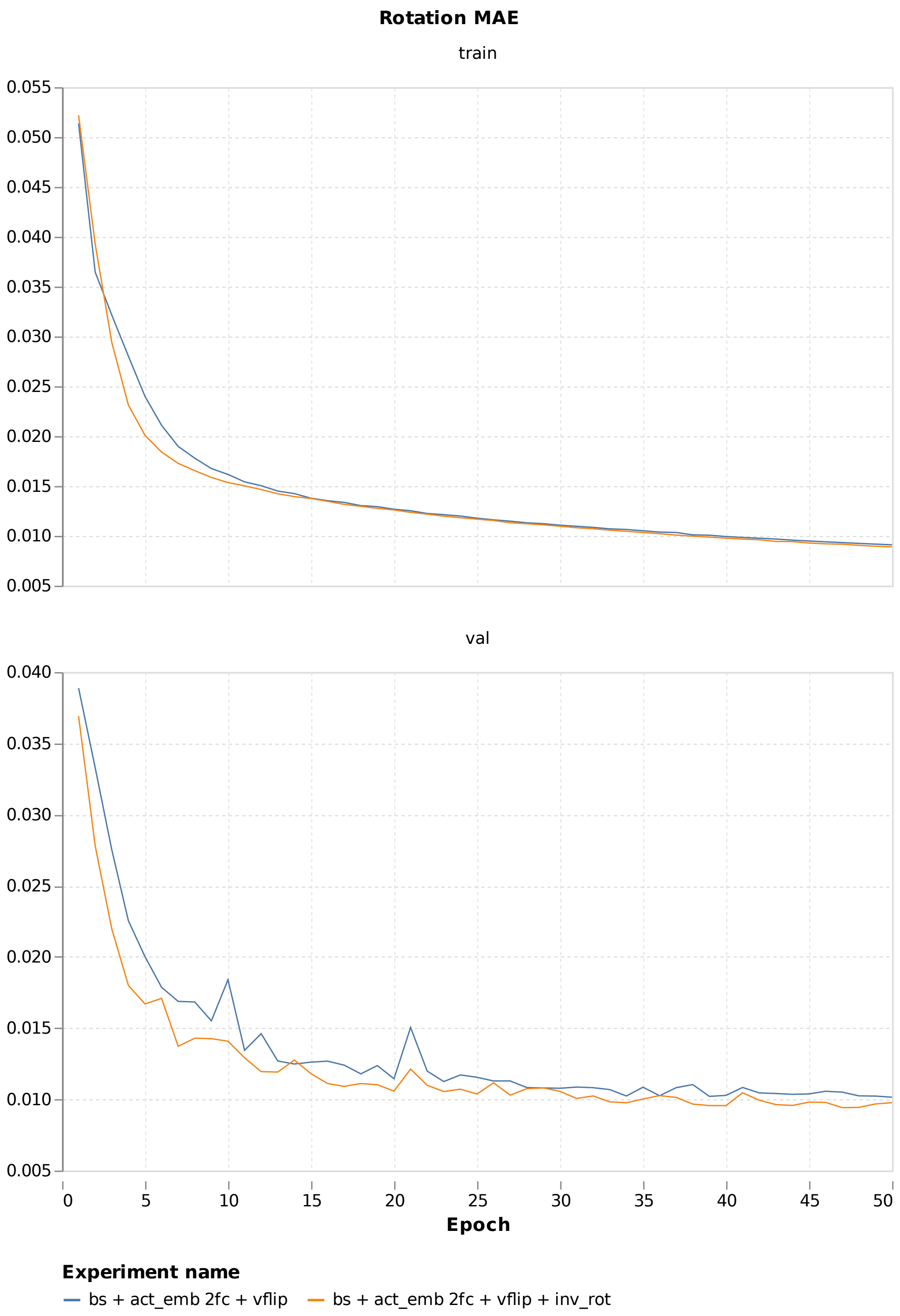


FIGURE 4.13: Frames swap: rotation MAE.

4.1.5 Separate Model per Each Action Type

In this section we analyze how the MAE changes if we train separate model (sepact) for estimating egomotion for each type of actions separately. We use the model (bs + act_emb 2fc) from Subsection 4.1.1 as a baseline.

Following experiments were launched:

- separate model for MOVE_FORWARD (bs + sepact_fwd);
- separate model for TURN_LEFT (sepact_left);
- separate model for TURN_RIGHT (sepact_right).

TABLE 4.5: Sepact: visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs + act_emb 2fc	20	3.00	2.67	3.39	3.47	1.25	0.96	1.58	1.68
	40	2.89	2.43	3.41	3.58	1.16	0.82	1.59	1.63
sepact_right	20	3.49	NA	NA	3.49	1.73	NA	NA	1.73
	40	3.46	NA	NA	3.46	1.71	NA	NA	1.71
sepact_left	20	3.34	NA	3.34	NA	1.76	NA	1.76	NA
	40	3.27	NA	3.27	NA	1.55	NA	1.55	NA
sepact_fwd	20	2.75	2.75	NA	NA	0.97	0.97	NA	NA
	40	2.27	2.27	NA	NA	0.77	0.77	NA	NA

In the table above MAE metrics for (bs + act_emb 2fc) experiment were computed on the whole validation dataset, but (sepact_right), (sepact_left) and (sepact_fwd) on the dataset items of a corresponding action type (also the plots in Figure 4.14 and Figure 4.15) so we can't directly compare these experiments. We show per-action metrics in the Appendix A.2.5. In figures in Appendix A.2.5 we may see that starting from the 25 epoch the sepact models usually have lower MAE than single model for all types of actions. It is worth mentioning that in this case we use 3 times more parameters than the baseline (single model).

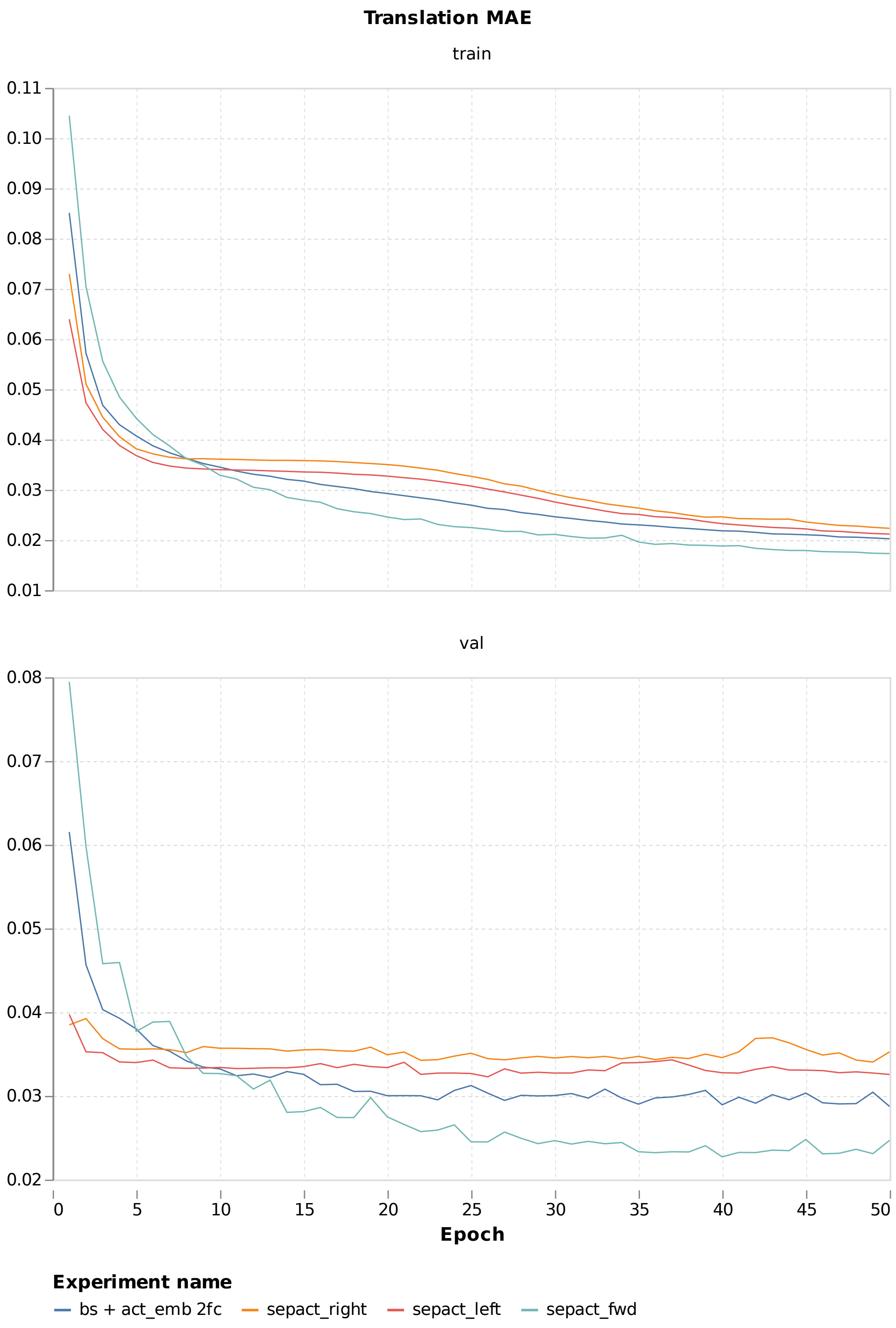


FIGURE 4.14: Separate model: translation MAE.

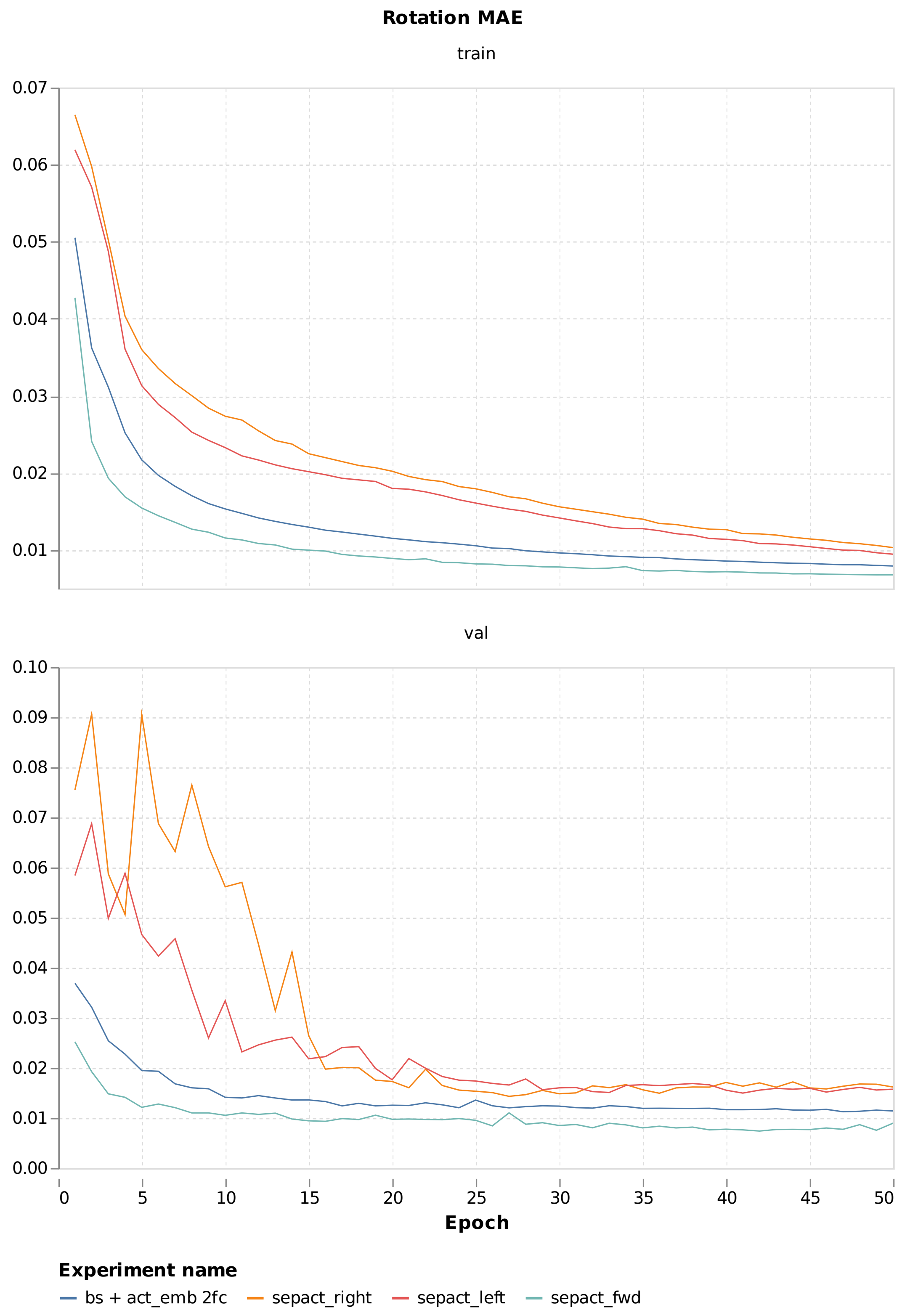


FIGURE 4.15: Separate model: rotation MAE.

4.1.6 Large Scale Training

We study the impact of training on the larger dataset consisting of 641052 pairs of observations in contrast to 130881 pairs of observations. We use the model (bs + act_emb 2fc) from Subsection 4.1.1 as a baseline.

Following experiments were launched:

- baseline + Large dataset (bs + act_emb 2fc + lscale).

TABLE 4.6: Large scale: visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs + act_emb 2fc	20	3.00	2.67	3.39	3.47	1.25	0.96	1.58	1.68
	40	2.89	2.43	3.41	3.58	1.16	0.82	1.59	1.63
bs + act_emb 2fc + lscale	20	2.37	1.88	2.97	3.03	0.82	0.53	1.22	1.18
	40	2.21	1.65	2.91	2.95	0.77	0.48	1.15	1.15

Training on large dataset significantly decreases translation and rotation MAE. We believe that by increasing dataset size more the model robustness can be enhanced further.

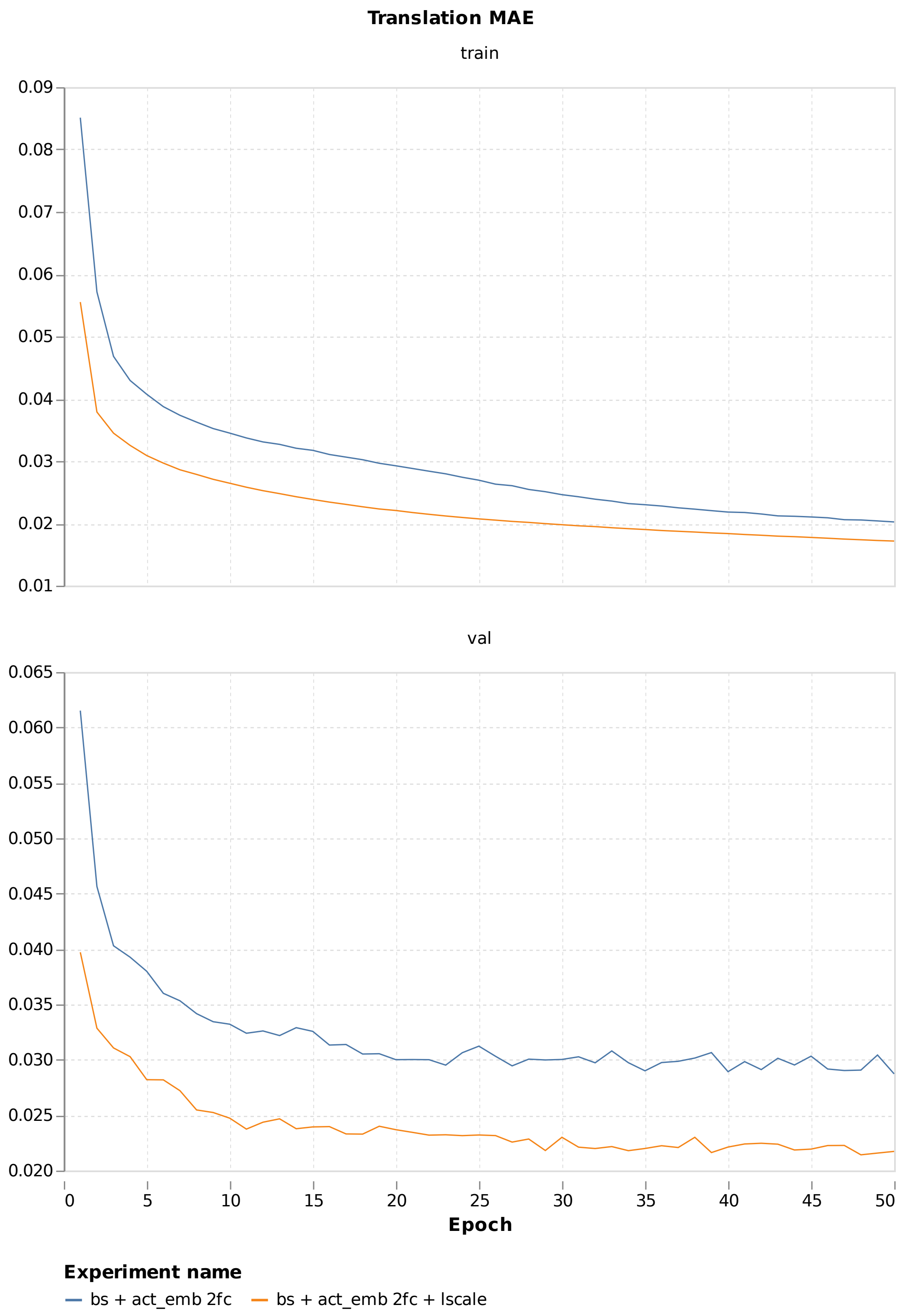


FIGURE 4.16: Large scale training: translation MAE.

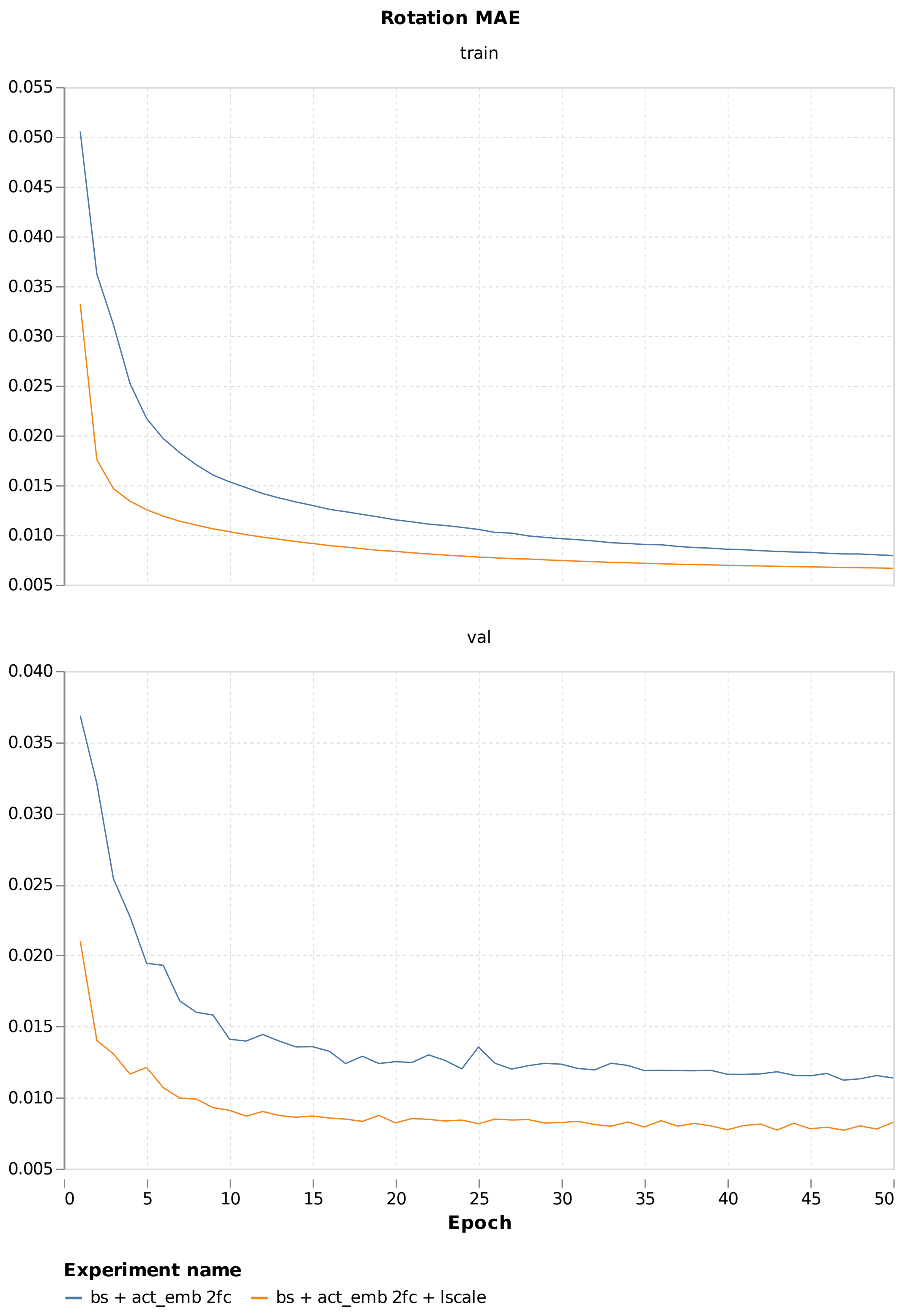


FIGURE 4.17: Large scale training: rotation MAE.

4.1.7 Summary

We summarize all the above experiments in one table below.

TABLE 4.7: Visual odometry metrics (subject to 1e+2 multiplication).

Experiment name	Epoch	Translation MAE				Rotation MAE			
		Total	Forward	Left	Right	Total	Forward	Left	Right
bs	20	4.61	5.41	3.48	3.67	2.37	1.58	3.61	3.17
	40	4.10	4.53	3.47	3.65	1.85	1.29	2.65	2.49
bs + col_emb	20	5.51	6.99	3.48	3.73	2.98	2.15	3.90	4.18
	40	4.65	5.45	3.56	3.67	2.40	1.64	3.28	3.45
bs + act_emb	20	3.13	2.86	3.45	3.48	1.32	0.99	1.74	1.76
	40	2.89	2.39	3.43	3.65	1.15	0.78	1.62	1.65
bs + col_emb + act_emb	20	3.10	2.80	3.37	3.62	1.37	1.02	1.79	1.87
	40	3.00	2.56	3.48	3.65	1.26	0.84	1.74	1.85
bs + act_emb 2fc	20	3.00	2.67	3.39	3.47	1.25	0.96	1.58	1.68
	40	2.89	2.43	3.41	3.58	1.16	0.82	1.59	1.63
bs + col_emb + act_emb 2fc	20	3.08	2.80	3.38	3.49	1.24	0.94	1.59	1.65
	40	2.87	2.43	3.31	3.57	1.17	0.85	1.55	1.60
bs + act_emb 2fc + vflip	20	2.92	2.54	3.37	3.44	1.14	0.85	1.51	1.53
	40	2.73	2.30	3.23	3.32	1.03	0.73	1.41	1.41
bs + act_emb 2fc + vflip + inv_rot	20	2.89	2.62	3.20	3.30	1.06	0.82	1.37	1.37
	40	2.76	2.42	3.17	3.23	0.96	0.69	1.28	1.32
bs + act_emb 2fc + 320x450->160x225	20	3.28	3.41	3.03	3.19	1.24	1.09	1.40	1.49
	40	3.14	3.18	3.01	3.16	1.15	0.95	1.38	1.44
bs + act_emb 2fc + 320x450->180x320	20	3.12	3.15	3.02	3.16	1.26	1.02	1.57	1.57
	40	2.94	2.89	3.00	3.00	1.18	1.00	1.41	1.42
bs + act_emb 2fc + 320x350->180x320	20	3.17	3.58	2.59	2.67	1.18	1.14	1.19	1.27
	40	2.76	3.05	2.36	2.41	0.98	0.95	1.02	1.00
sepact_right	20	3.49	NA	NA	3.49	1.73	NA	NA	1.73
	40	3.46	NA	NA	3.46	1.71	NA	NA	1.71
sepact_left	20	3.34	NA	3.34	NA	1.76	NA	1.76	NA
	40	3.27	NA	3.27	NA	1.55	NA	1.55	NA
sepact_fwd	20	2.75	2.75	NA	NA	0.97	0.97	NA	NA
	40	2.27	2.27	NA	NA	0.77	0.77	NA	NA
bs + act_emb 2fc + lscale	20	2.37	1.88	2.97	3.03	0.82	0.53	1.22	1.18
	40	2.21	1.65	2.91	2.95	0.77	0.48	1.15	1.15

In the table above we may see that prediction errors are not uniformly distributed across action types. The experiments that have the same MAE on the same validation dataset may have completely different error distribution across actions (split of the dataset that corresponds to a particular action type). So it makes model analysis easier when per-action metrics are computed. Our experiments show that all models except ones that were trained on the observation crops tend to learn translation distribution better than rotation (even the model with inversed rotations added that has almost uniform actions distribution).

4.2 Navigation

We replaced the policy GRS+Compass signal with the output of our visual odometry module and report the navigation metrics for models that has lowest MAE in each experiments group from Section 4.1 in table below. No further fine-tuning was applied after replacing the ground-truth signal. We launched two navigation policies: one with standard success zone radius - 0.36 meters and the other with narrowed success zone radius - 0.18 meters assuming that smaller radius will force the agent to call STOP action more accurately. To study the impact of different visual odometry modules we fixed the navigation policies (used the same network weights) across all experiments.

4.2.1 Summary

TABLE 4.8: Navigation metrics.

Experiment name	Epoch	TTA		Success region	Navigation metrics			
		Flip	Swap		SPL	SoftSPL	Success	Distance to goal
bs	20	-	-	0.36	0.276	0.606	0.365	1.392
	40	-	-	0.36	0.307	0.612	0.408	1.235
bs + act_emb 2fc	20	-	-	0.36	0.477	0.693	0.615	0.676
	40	-	-	0.36	0.470	0.690	0.599	0.761
bs + act_emb 2fc	20	+	-	0.36	0.504	0.696	0.648	0.706
	40	+	-	0.36	0.524	0.714	0.665	0.623
bs + act_emb 2fc	20	-	+	0.36	0.513	0.708	0.663	0.514
	40	-	+	0.36	0.483	0.697	0.618	0.670
bs + act_emb 2fc	20	+	+	0.36	0.529	0.706	0.676	0.585
	40	+	+	0.36	0.506	0.709	0.638	0.661
bs + act_emb 2fc + vflip	20	-	-	0.36	0.460	0.687	0.596	0.707
	40	-	-	0.36	0.552	0.699	0.717	0.595
bs + act_emb 2fc + vflip	20	+	-	0.36	0.529	0.706	0.686	0.544
	40	+	-	0.36	0.548	0.701	0.708	0.531
bs + act_emb 2fc + vflip + inv_rot	20	-	-	0.36	0.548	0.703	0.709	0.522
	40	-	-	0.36	0.525	0.695	0.678	0.573
bs + act_emb 2fc + vflip + inv_rot	20	+	+	0.36	0.559	0.710	0.716	0.515
	40	+	+	0.36	0.573	0.697	0.746	0.528
bs + act_emb 2fc + 320x450->180x320	20	-	-	0.36	0.496	0.701	0.634	0.690
	40	-	-	0.36	0.449	0.695	0.575	0.704
bs + act_emb 2fc + 320x350->180x320	20	-	-	0.36	0.472	0.695	0.601	0.729
	40	-	-	0.36	0.547	0.709	0.701	0.480
sepact	20	-	-	0.36	0.470	0.688	0.604	0.658
	40	-	-	0.36	0.527	0.694	0.684	0.600
bs + act_emb 2fc + lscale	20	-	-	0.36	0.587	0.711	0.760	0.484
	40	-	-	0.36	0.599	0.707	0.780	0.427
bs	20	-	-	0.18	0.265	0.591	0.353	1.502
	40	-	-	0.18	0.278	0.591	0.381	1.313
bs + act_emb 2fc	20	-	-	0.18	0.461	0.671	0.604	0.748
	40	-	-	0.18	0.473	0.681	0.609	0.776
bs + act_emb 2fc + lscale	20	-	-	0.18	0.548	0.682	0.720	0.586
	40	-	-	0.18	0.558	0.672	0.741	0.590

Our experiments show that all the regularization techniques proposed in Section 4.1 decrease the MAE improving navigation performance comparing to the baseline. Moreover, they can be combined together. Flip and swap dataset augmentations may be applied not only at train-time but also at evaluation-time (navigation). It acts as a regularization and improves visual odometry module robustness at almost no cost. According to our experiments results, large and diverse dataset training is one of the factors that contributes to the robust navigation the most. When we increased the training dataset size from 130881 pairs of observations to 641052 pairs of observations (approximately 5 times increase) the navigation metrics improved significantly (+0.11 SPL, +0.145 Success on epoch 20 and +0.129SPL, +0.181 Success on epoch 40).The policy with the standard success zone radius has better navigation metrics than the policy with narrowed success zone radius.

4.2.2 Habitat Challenge 2021 Benchmark

TABLE 4.9: Habitat Challenge 2021 public leaderboard (06.06.2021).

Participant team	SPL	SoftSPL	Success	Distance to goal
Ours	**0.66**	**0.71**	**0.86**	**0.61**
Differentiable SLAM-net	0.44	0.58	0.63	1.80
Habitat Team (RGBD+DD-PPO) (baseline)	0.00	0.13	0.00	5.84

Our navigation agent consisting of policy trained with standard success zone radius and visual odometry module trained in the (bs + act_emb 2fc + vflip + inv_rot) setup has shown competitive results in the Habitat Challenge 2021 benchmark (see Table 4.9). The model was trained by Oleksandr Maksymets on the Facebook AI Research cluster. Training dataset was collected by following protocol described in Subsection 3.3.2 and consists of more than 2 million observation frames (with the same actions distribution as shown in Figure 3.6). Our method outperforms the state-of-the-art approach of the Habitat Challenge 2020 benchmark (+0.132 SPL and +0.142 Success).

Chapter 5

Conclusive Remarks

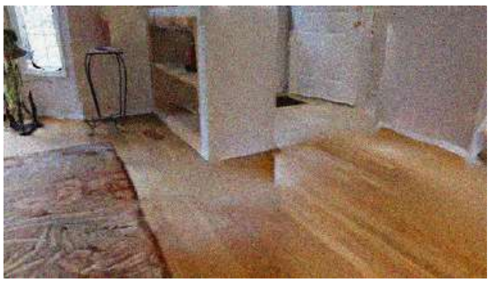
In this master thesis, we approached the problem of PointGoal navigation. We analyzed the related work in the field and outlined the research gaps in the form of the open research questions and formulated the relevant research hypotheses that were validated in our experiments. Proposed regularization techniques proved to improve the visual odometry module accuracy and result in more efficient navigation and robustness to actuation and sensing noise. Navigation agent equipped with our best visual odometry module scores competitive results (0.66 SPL and 0.86 Success, see Table 4.9) in Habitat Challenge 2021 benchmark even without policy and odometry fine-tuning. Our method outperforms the state-of-the-art approach of the Habitat Challenge 2020 benchmark (+0.132 SPL and +0.142 Success). We also show that the visual odometry model robustness to noise highly depends on the size and diversity of the training dataset. When we increased the training dataset size from 130881 pairs of observations to 641052 pairs of observations (approximately 5 times increase) the navigation metrics improved significantly (+0.11 SPL, +0.145 Success on epoch 20 and +0.129SPL, +0.181 Success on epoch 40, see Table 4.8). We think that the navigation efficiency may be improved further by incorporating new scenes and increasing the training dataset size. On the other hand, we noticed that to reach the competitive performance the size of the dataset may occupy a lot of space on the disk. For instance, $\sim$600000 observation pairs stored on disk occupy $\sim$1.3TB of disk space. And the dataset has to be regenerated every time the task specification changes. Moreover the epoch training time increases proportionally. But we believe that the computational and storage cost may be decreased by training visual odometry model online from the Habitat simulator.

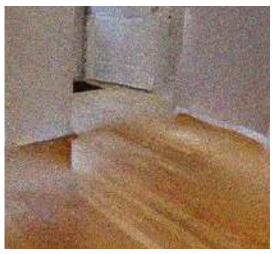
We see the end-to-end policy and visual odometry training where the visual odometry model is trained simultaneously with navigation policy (for instance using the same observations buffer) as a good direction for future research. Model generalization across different scene datasets is not studied well also (for instance, training on the Gibson dataset and validating on Matterport3D and vice versa).

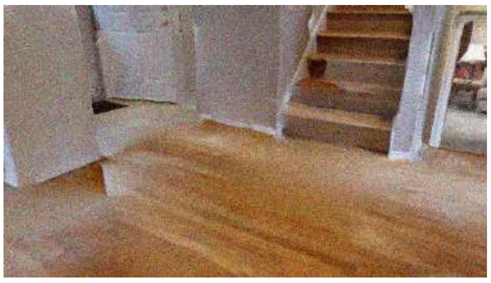
Appendix A

Visualizations

A.1 Visual Observations Intersection Region Crop


(A) Source RGB observation.


(B) Source RGB observation cropped.


(C) Target RGB observation.

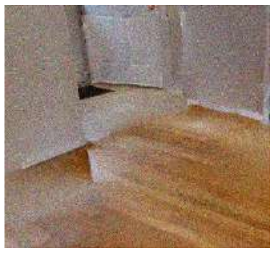
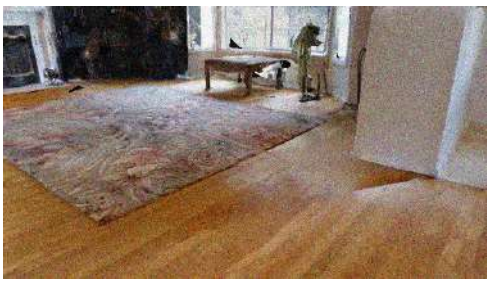
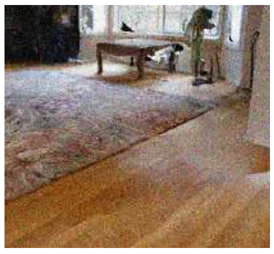

(D) Target RGB observation cropped.

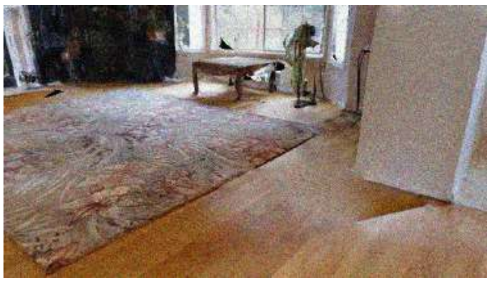
FIGURE A.1: TURN_RIGHT crop 320 × 350.



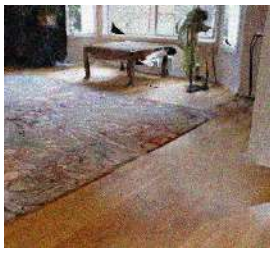
(A) Source RGB observation.



(B) Source RGB observation cropped.



(C) Target RGB observation.



(D) Target RGB observation cropped.

FIGURE A.2: MOVE_FORWARD crop 320 $\times$ 350.

A.2 Visual Odometry Per-action Metrics

A.2.1 Embeddings

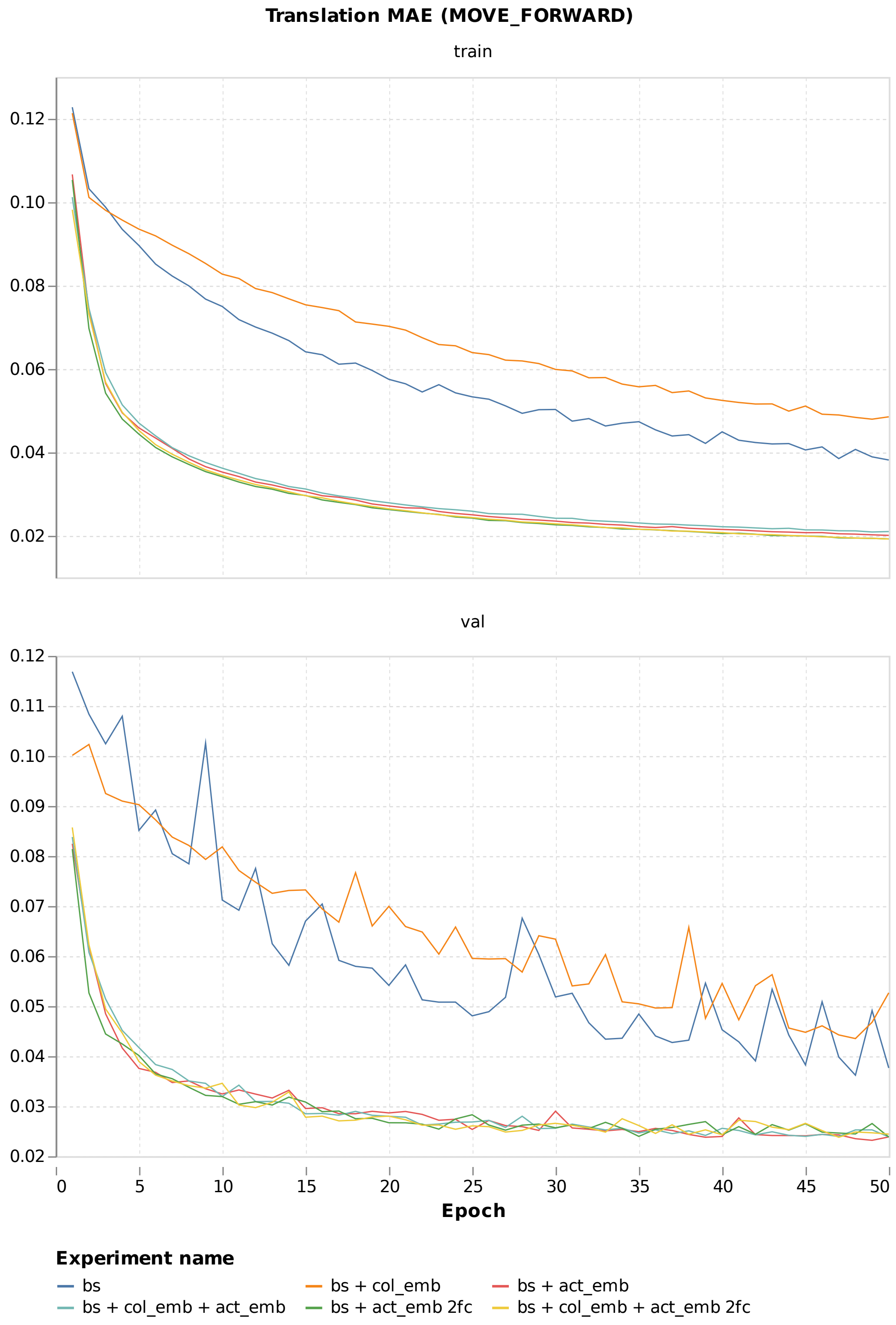


FIGURE A.3: Embeddings: translation MAE for MOVE_FORWARD action.

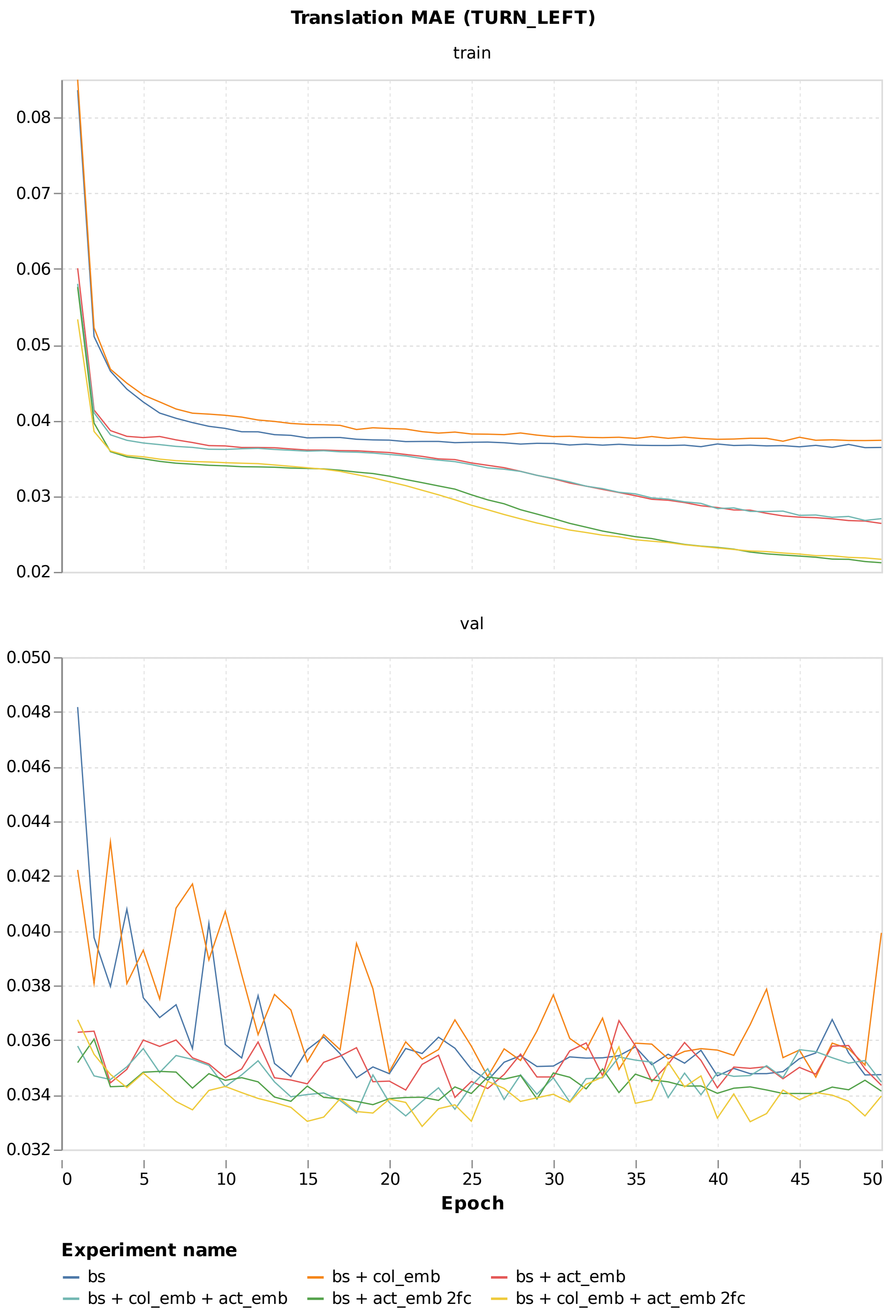


FIGURE A.4: Embeddings: translation MAE for TURN_LEFT action.

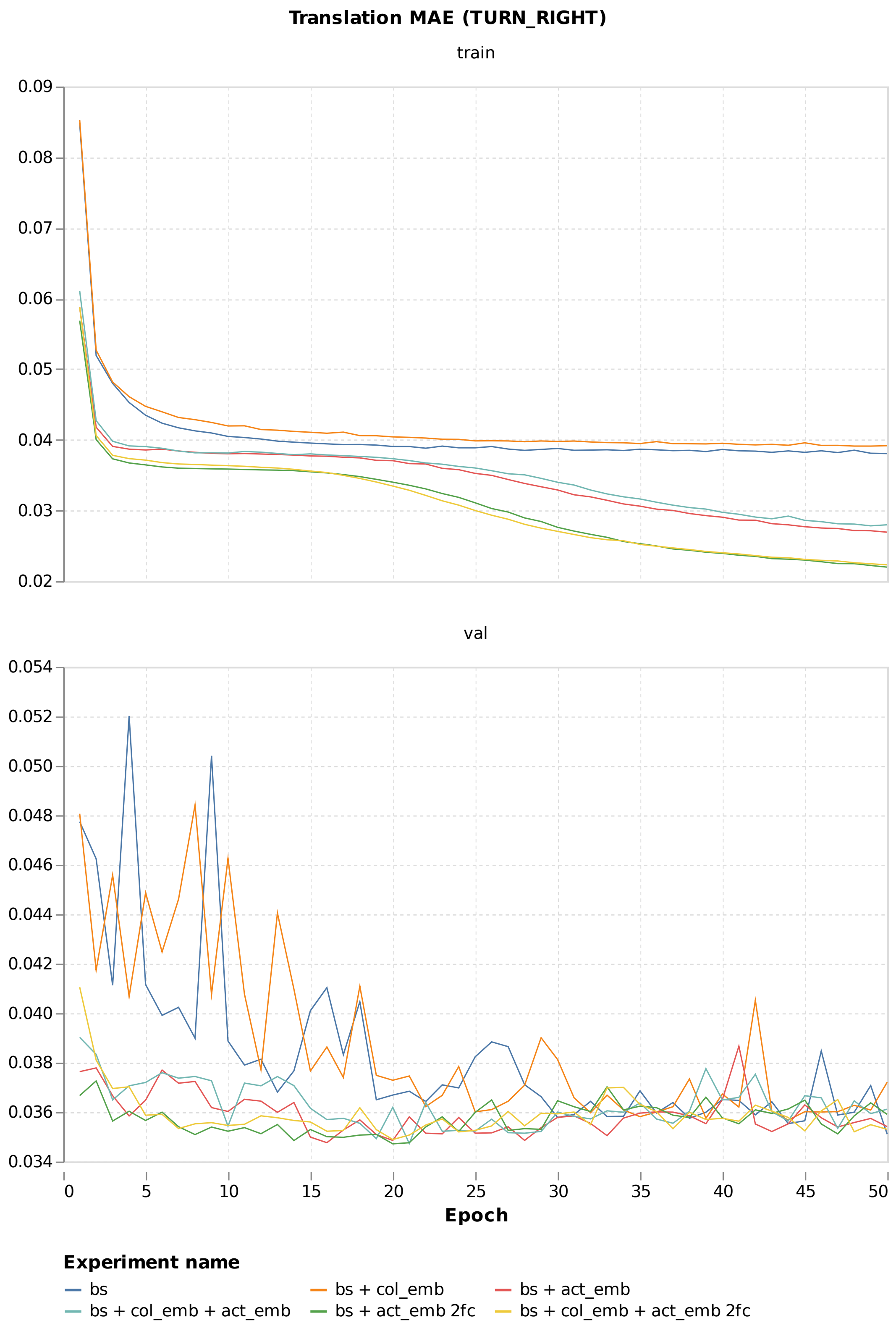


FIGURE A.5: Embeddings: translation MAE for TURN_RIGHT action.

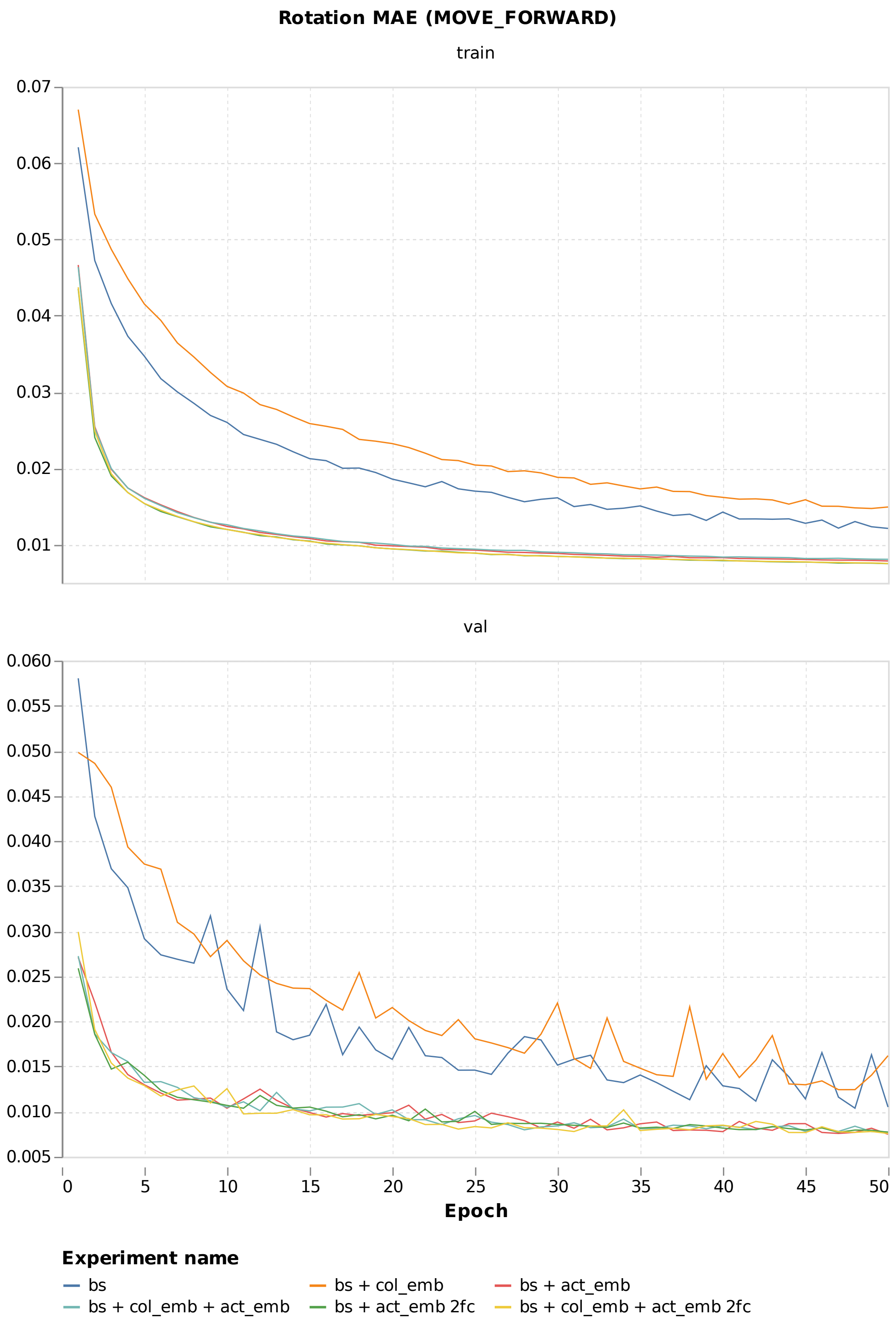


FIGURE A.6: Embeddings: rotation MAE for MOVE_FORWARD action.

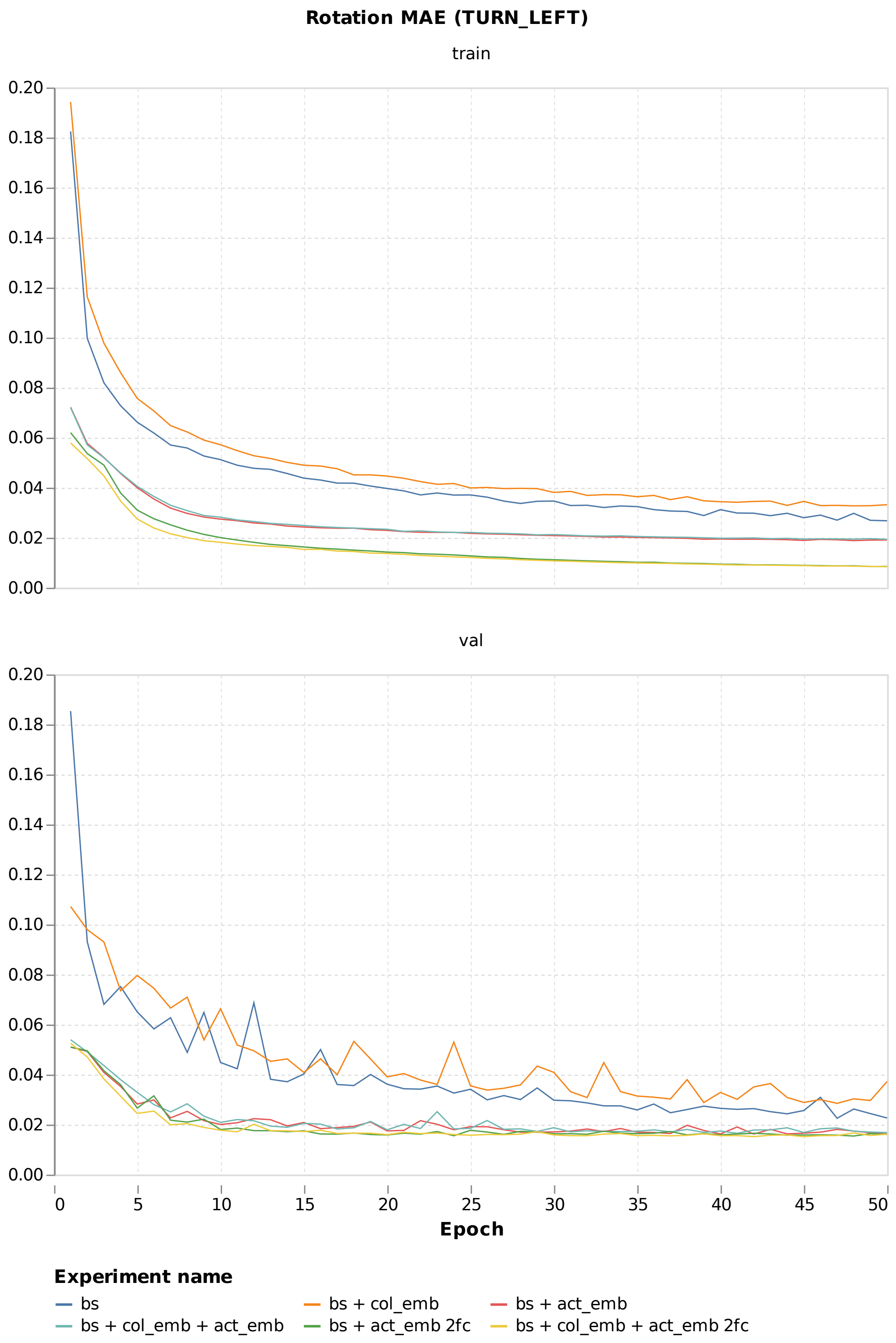


FIGURE A.7: Embeddings: rotation MAE for TURN_LEFT action.

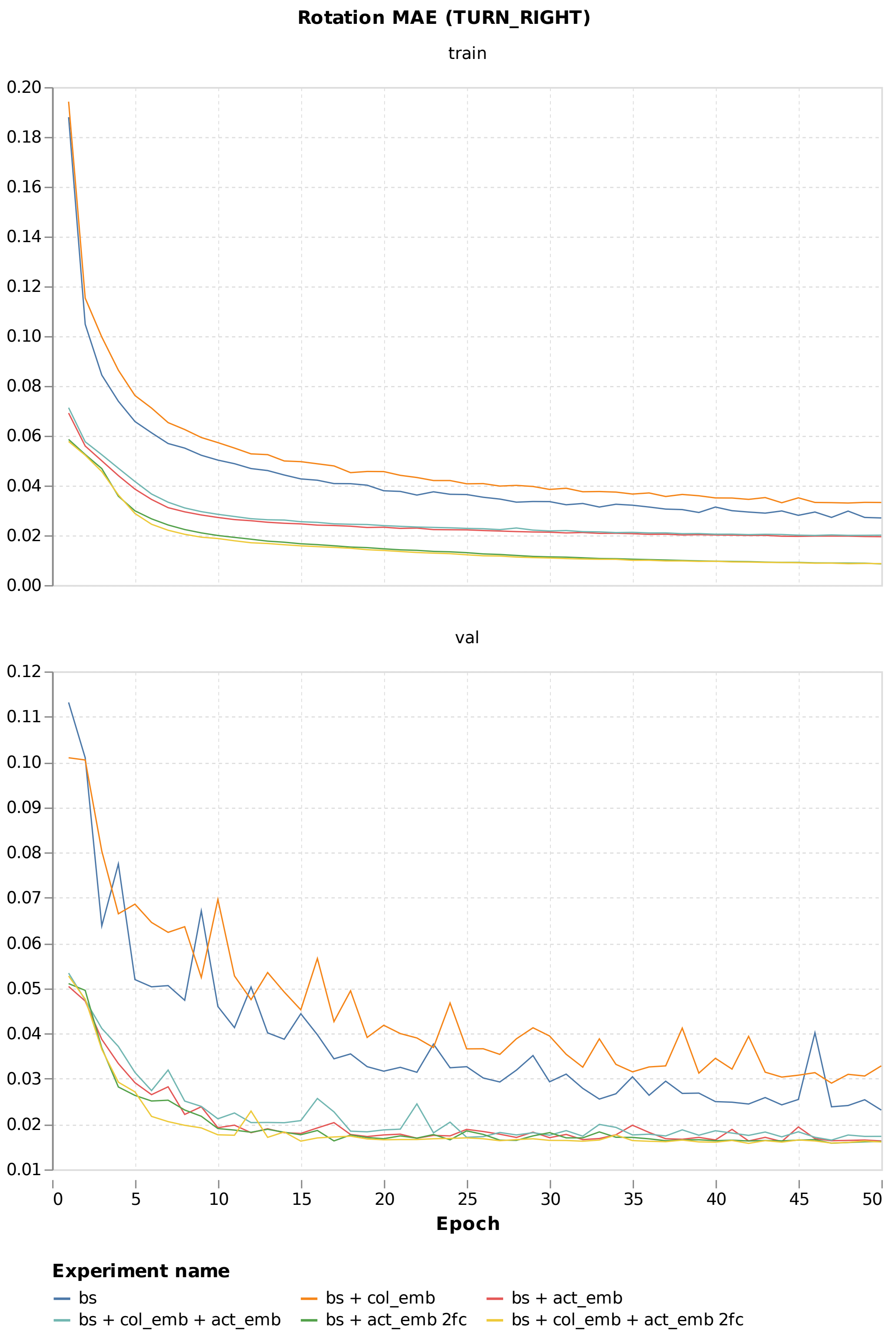


FIGURE A.8: Embeddings: rotation MAE for TURN_RIGHT action.

A.2.2 Visual Observations Intersection Region Crop

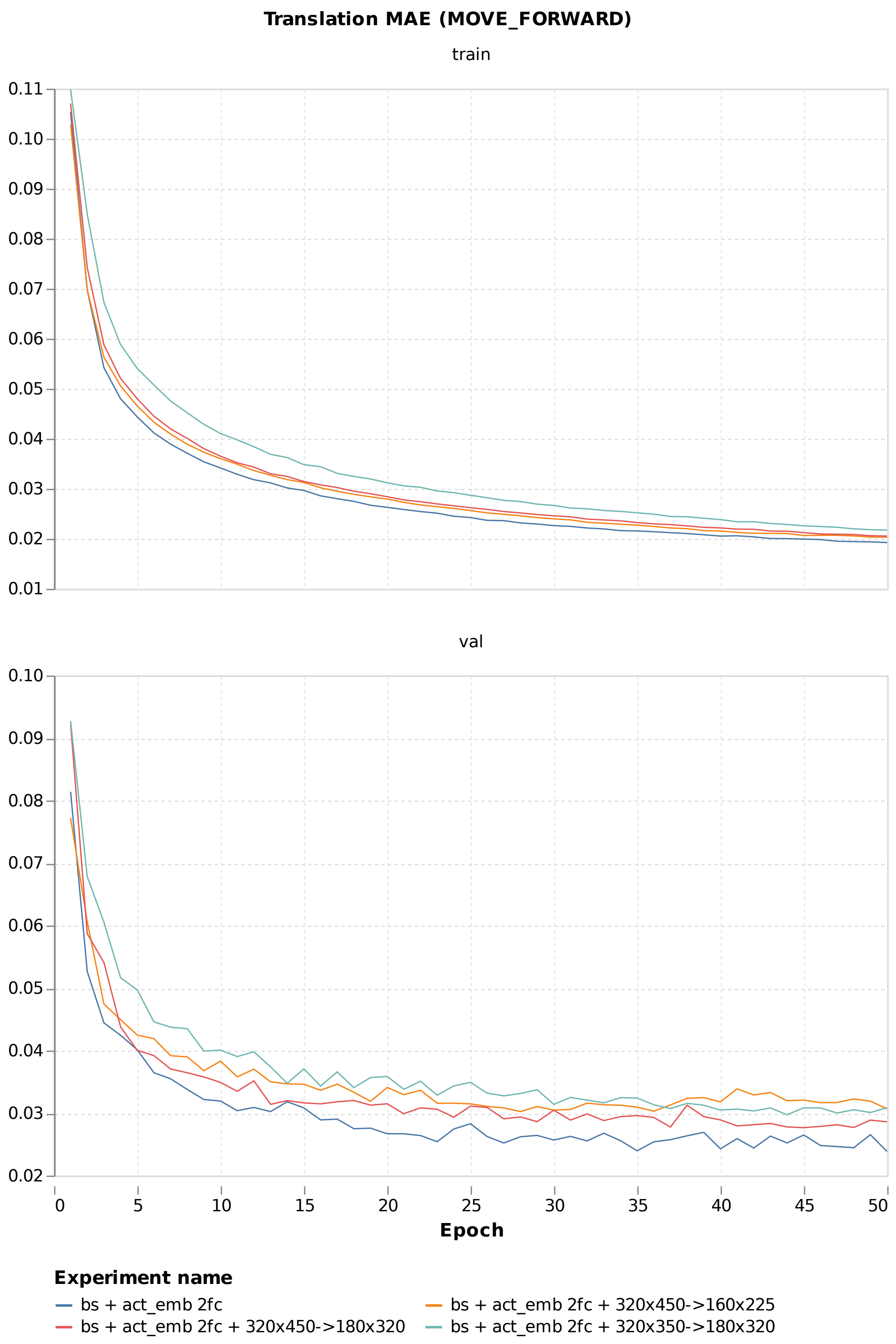


FIGURE A.9: Crop: translation MAE for MOVE_FORWARD action.

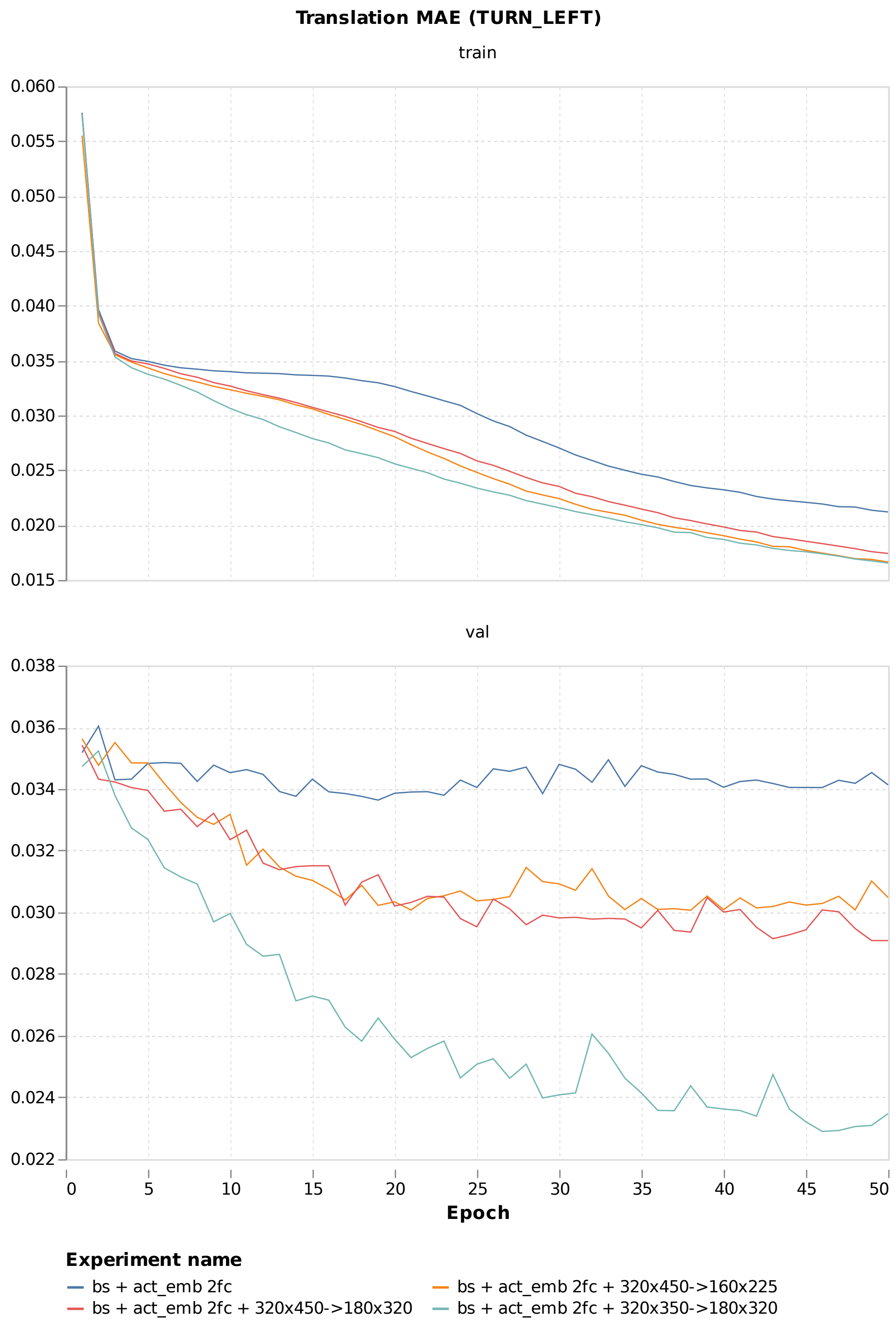


FIGURE A.10: Crop: translation MAE for TURN_LEFT action.

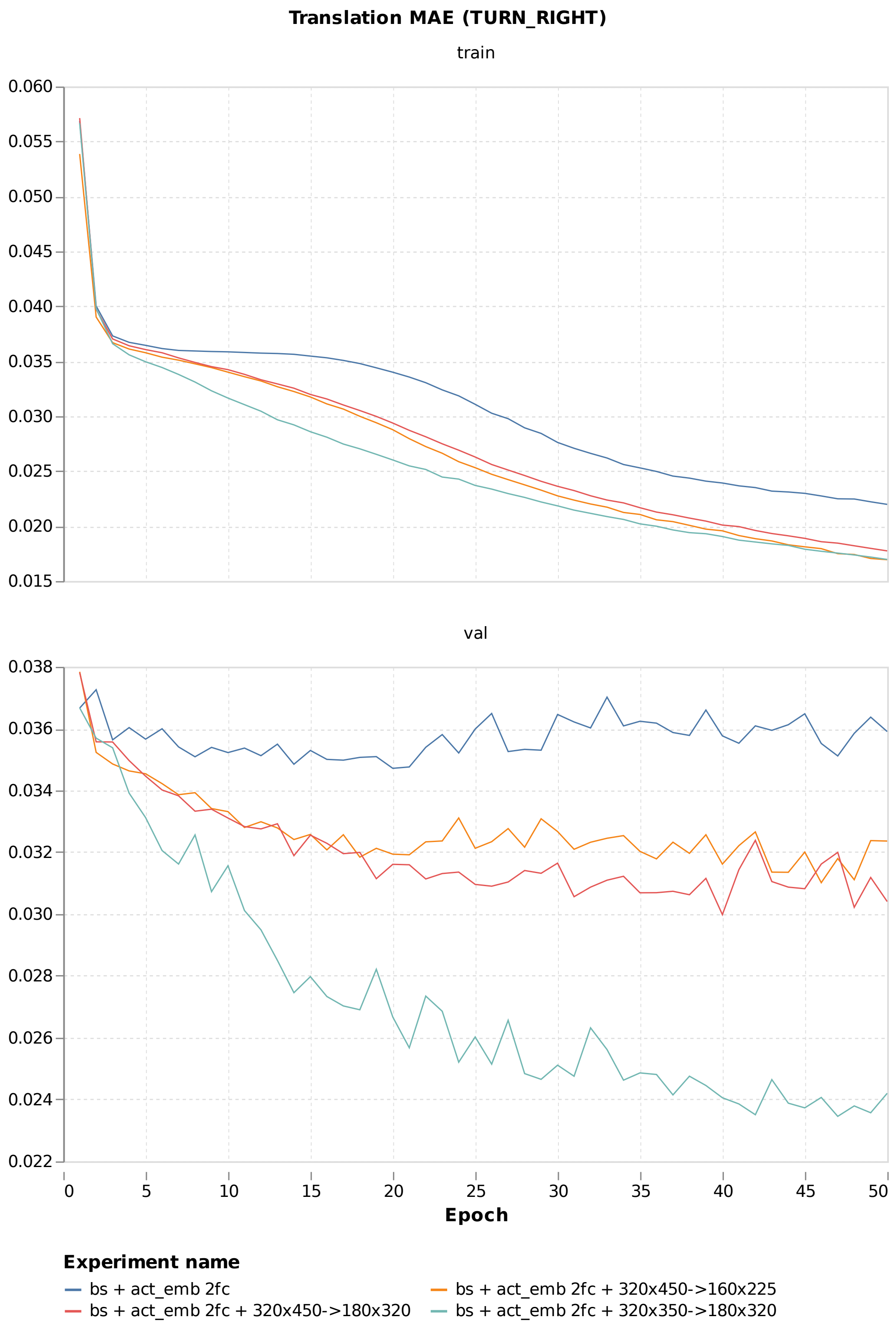


FIGURE A.11: Crop: translation MAE for TURN_RIGHT action.

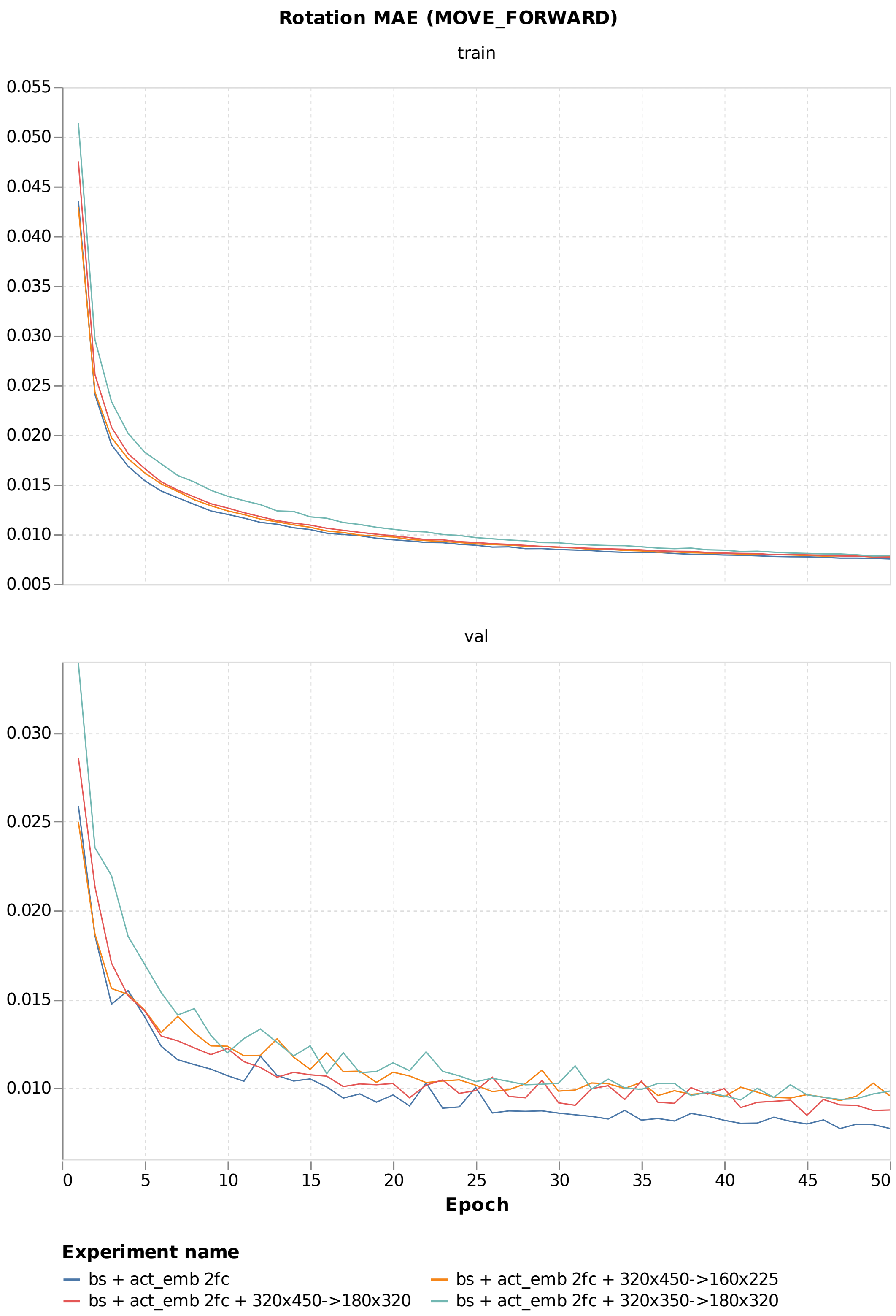


FIGURE A.12: Crop: rotation MAE for MOVE_FORWARD action.

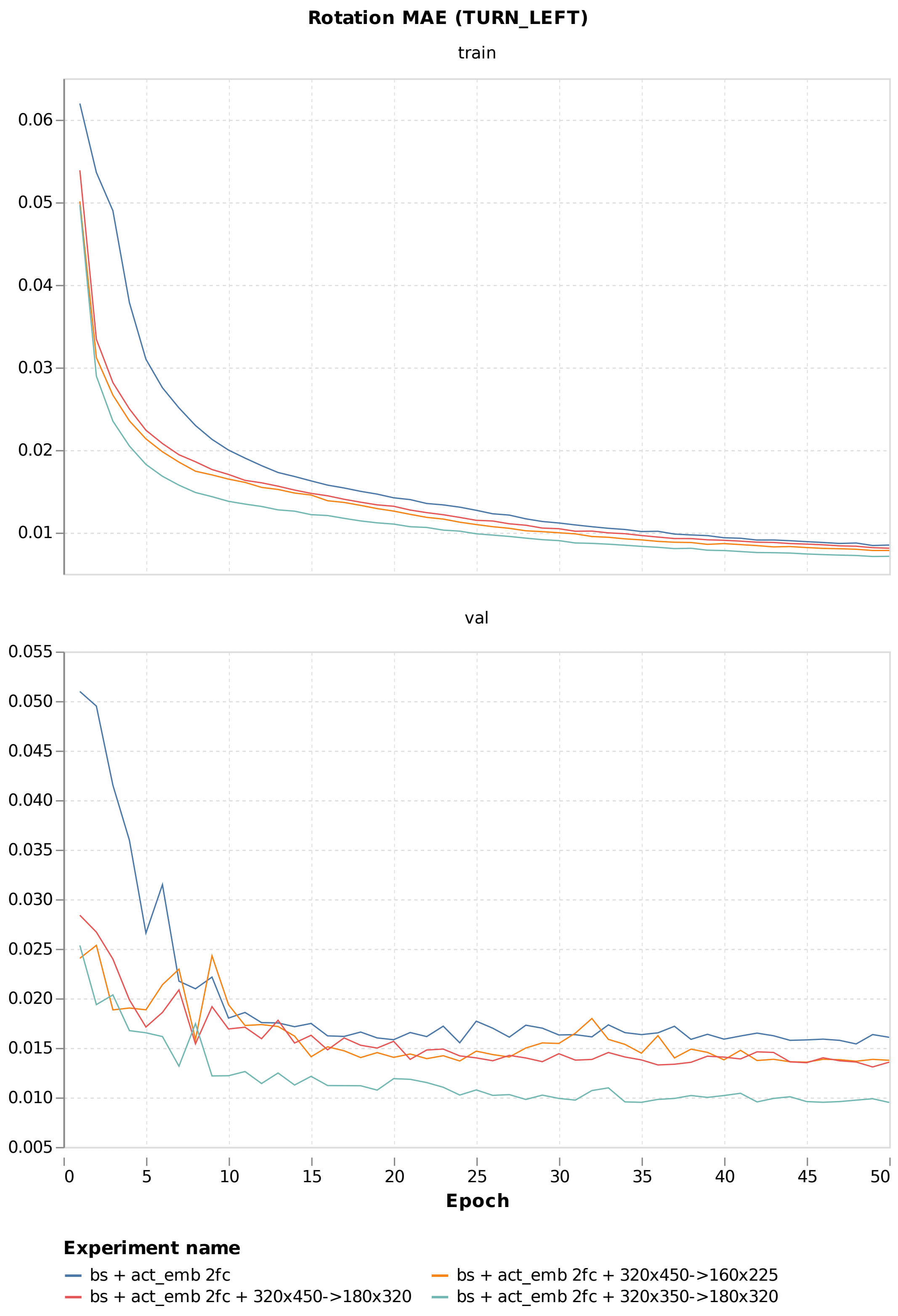


FIGURE A.13: Crop: rotation MAE for TURN_LEFT action.

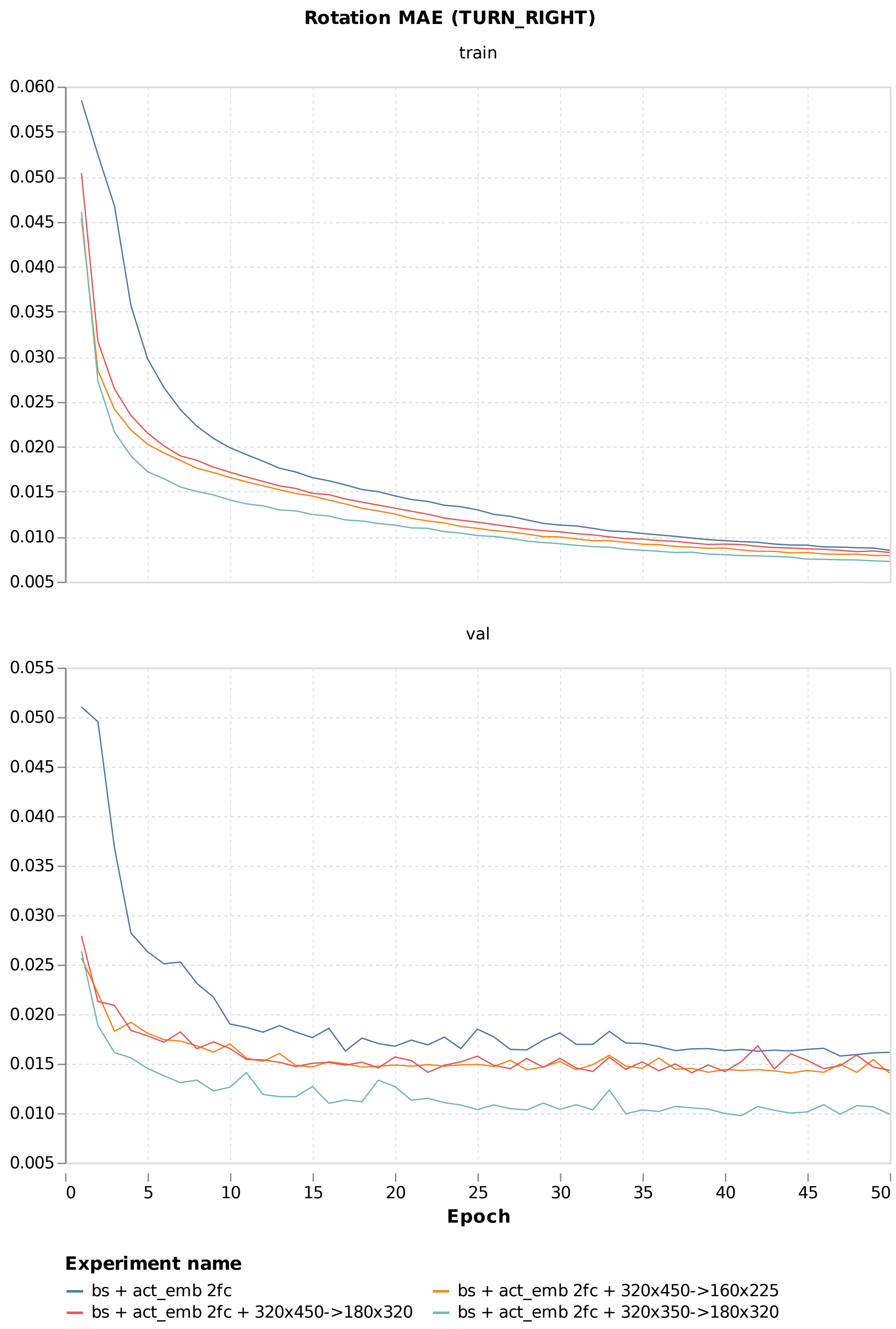


FIGURE A.14: Crop: rotation MAE for TURN_RIGHT action.

A.2.3 Frames Flip

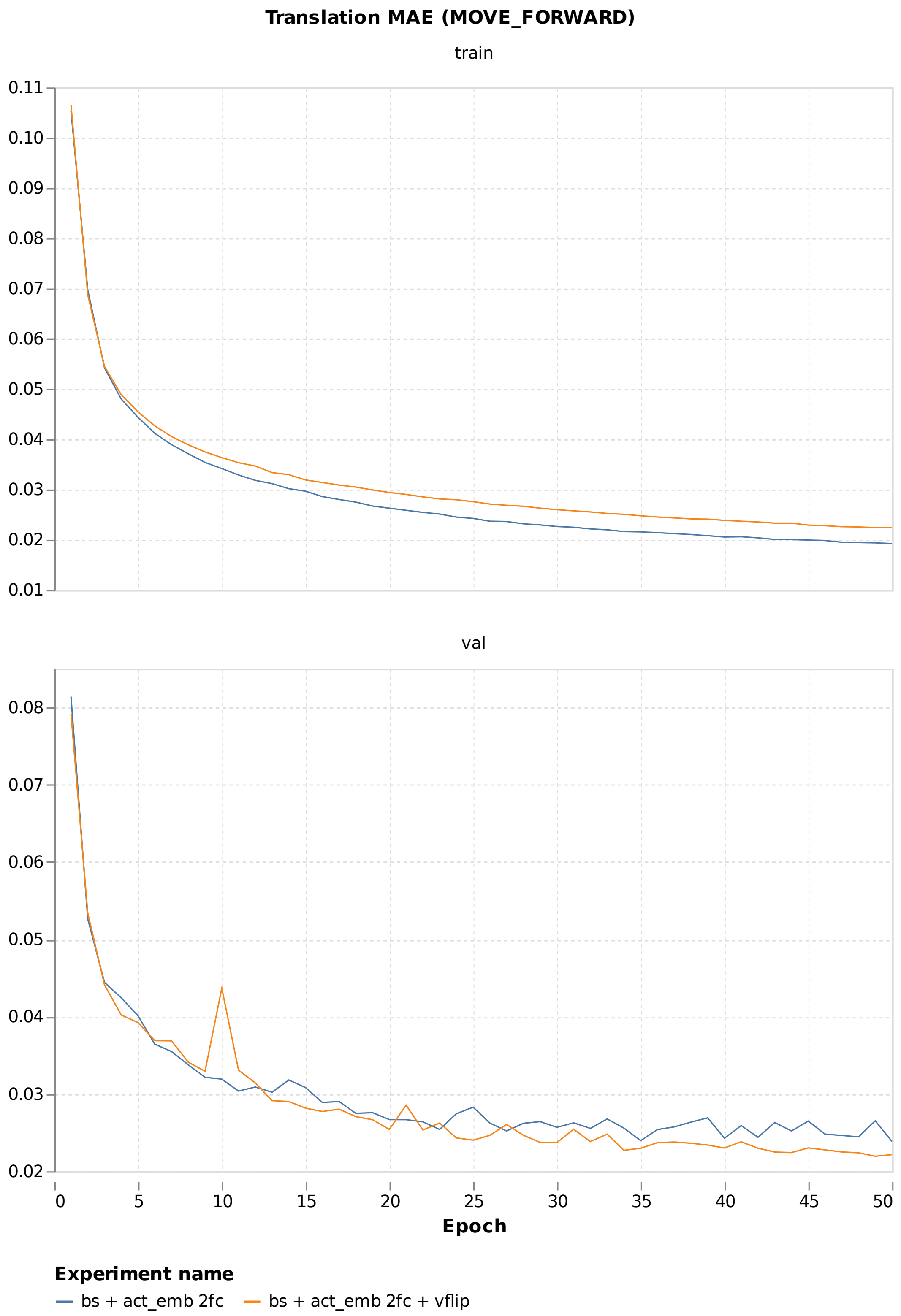


FIGURE A.15: Flip: translation MAE for MOVE_FORWARD action.

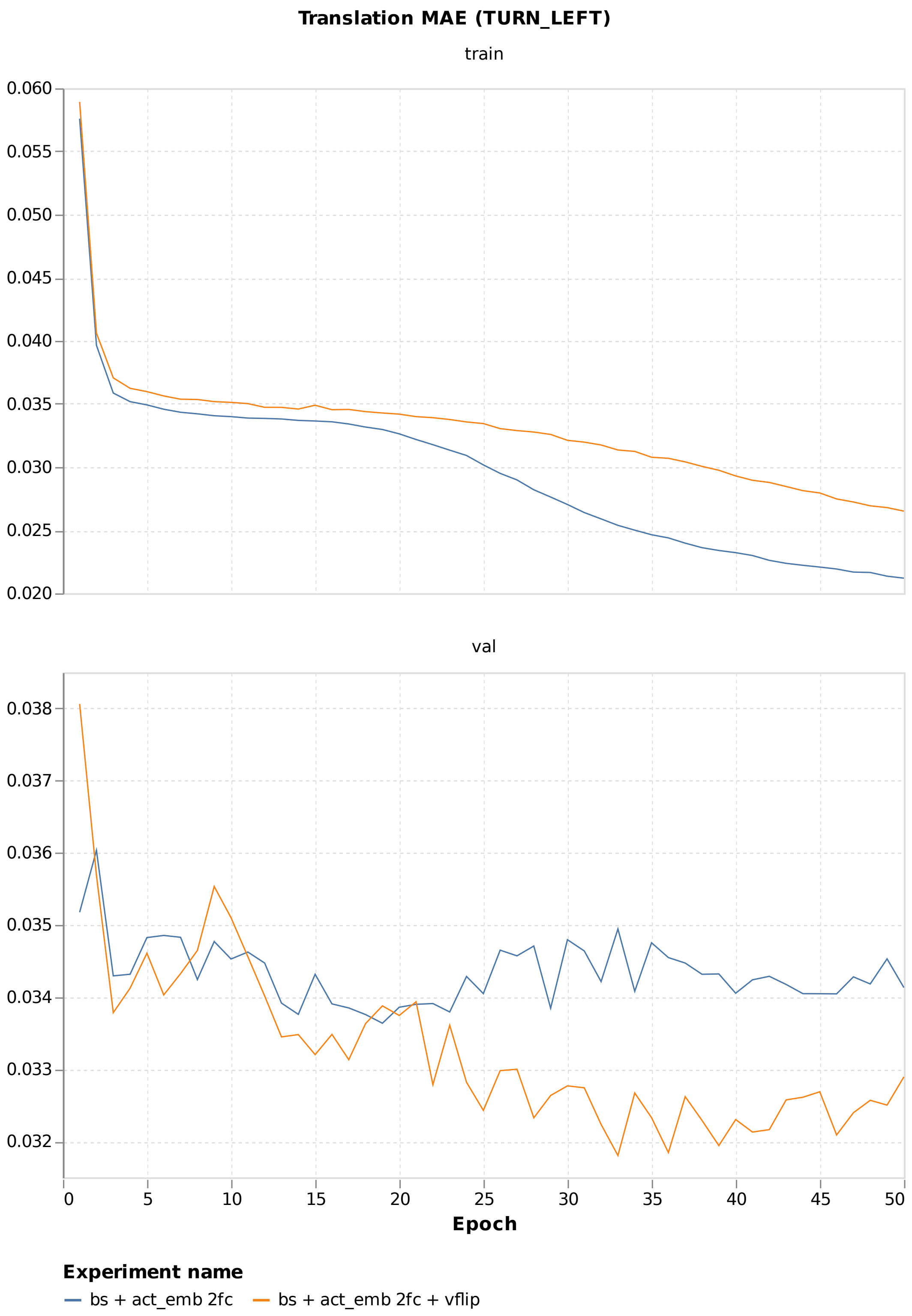


FIGURE A.16: Flip: translation MAE for TURN_LEFT action.

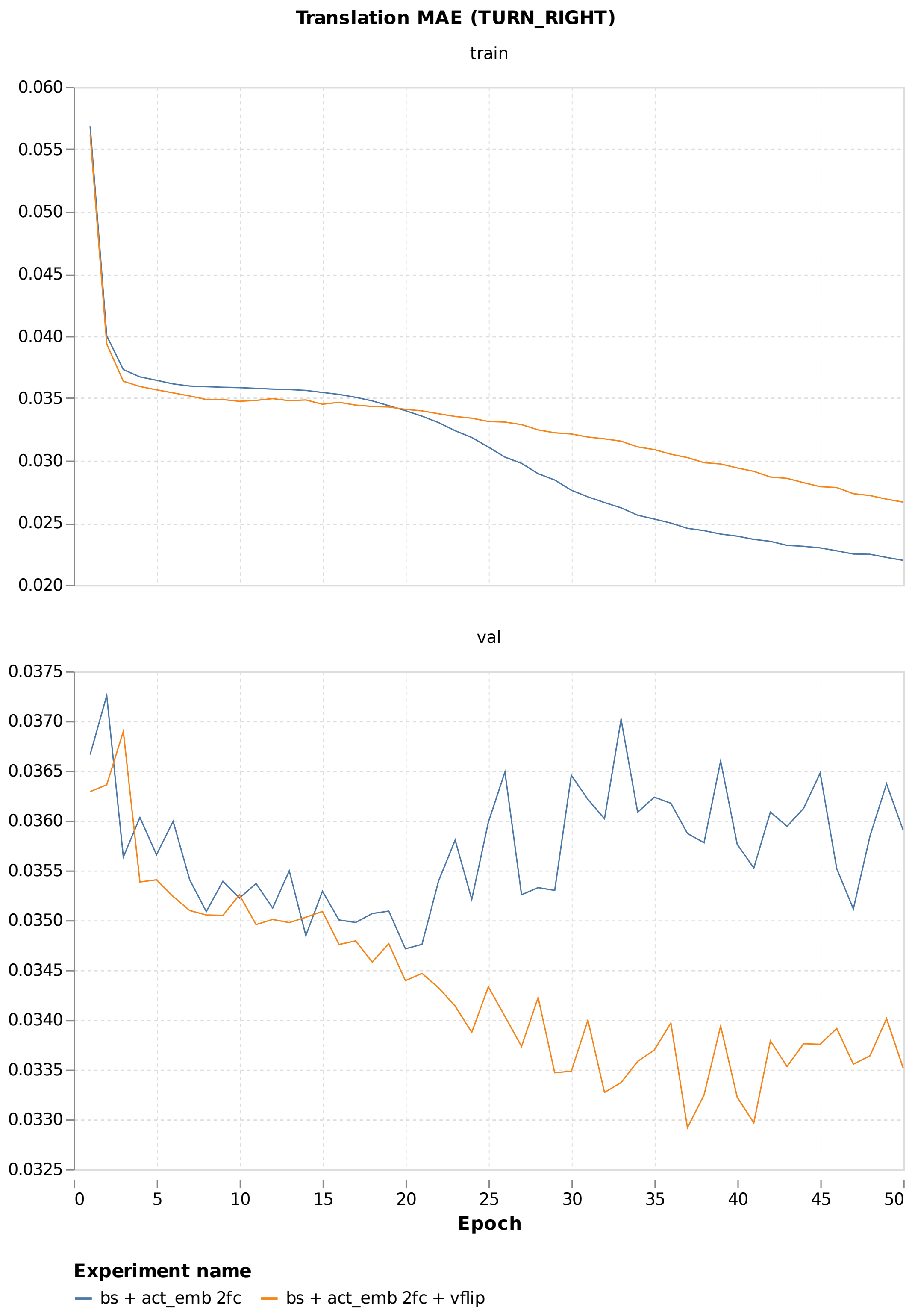


FIGURE A.17: Flip: translation MAE for TURN_RIGHT action.

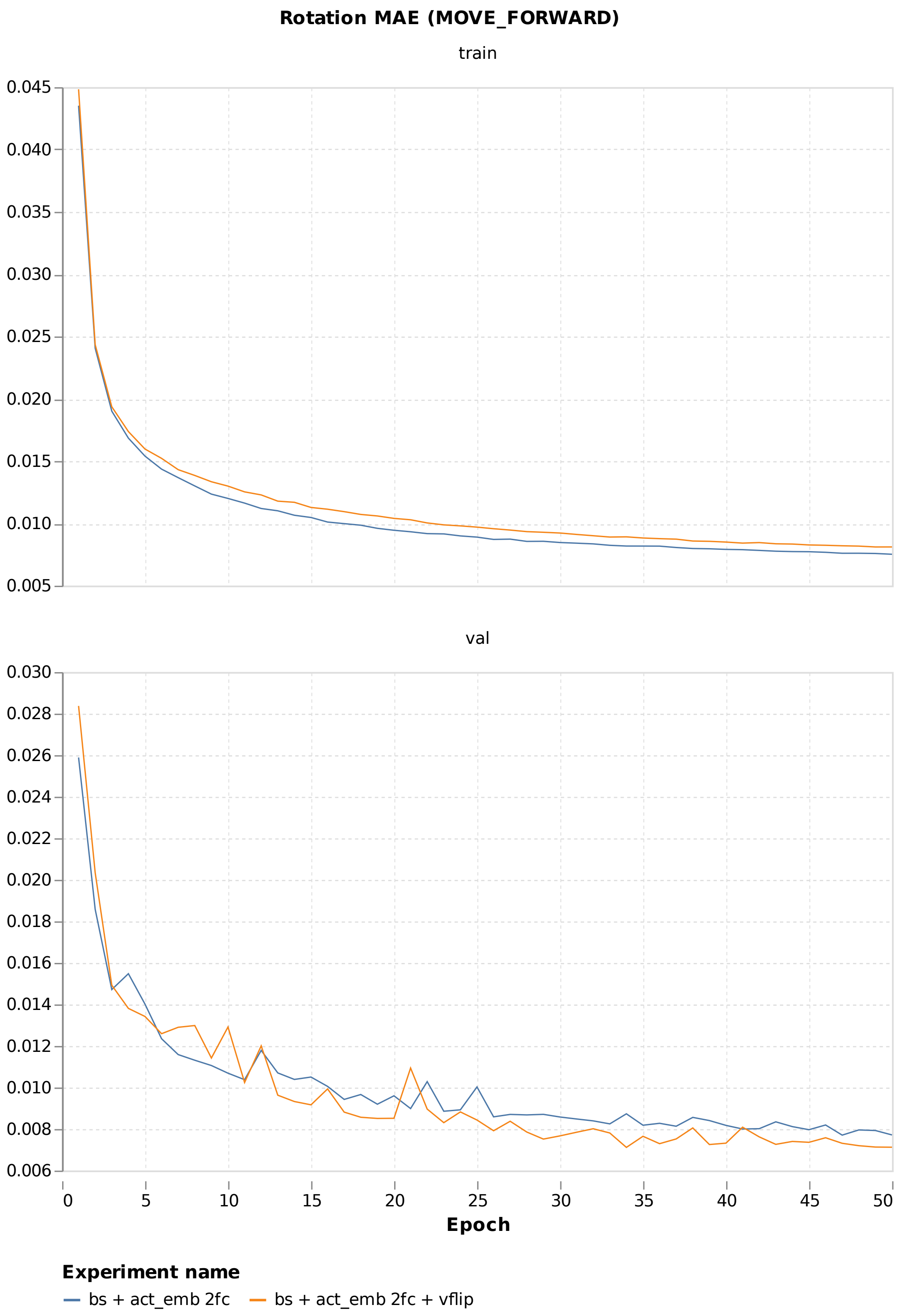


FIGURE A.18: Flip: rotation MAE for MOVE_FORWARD action.

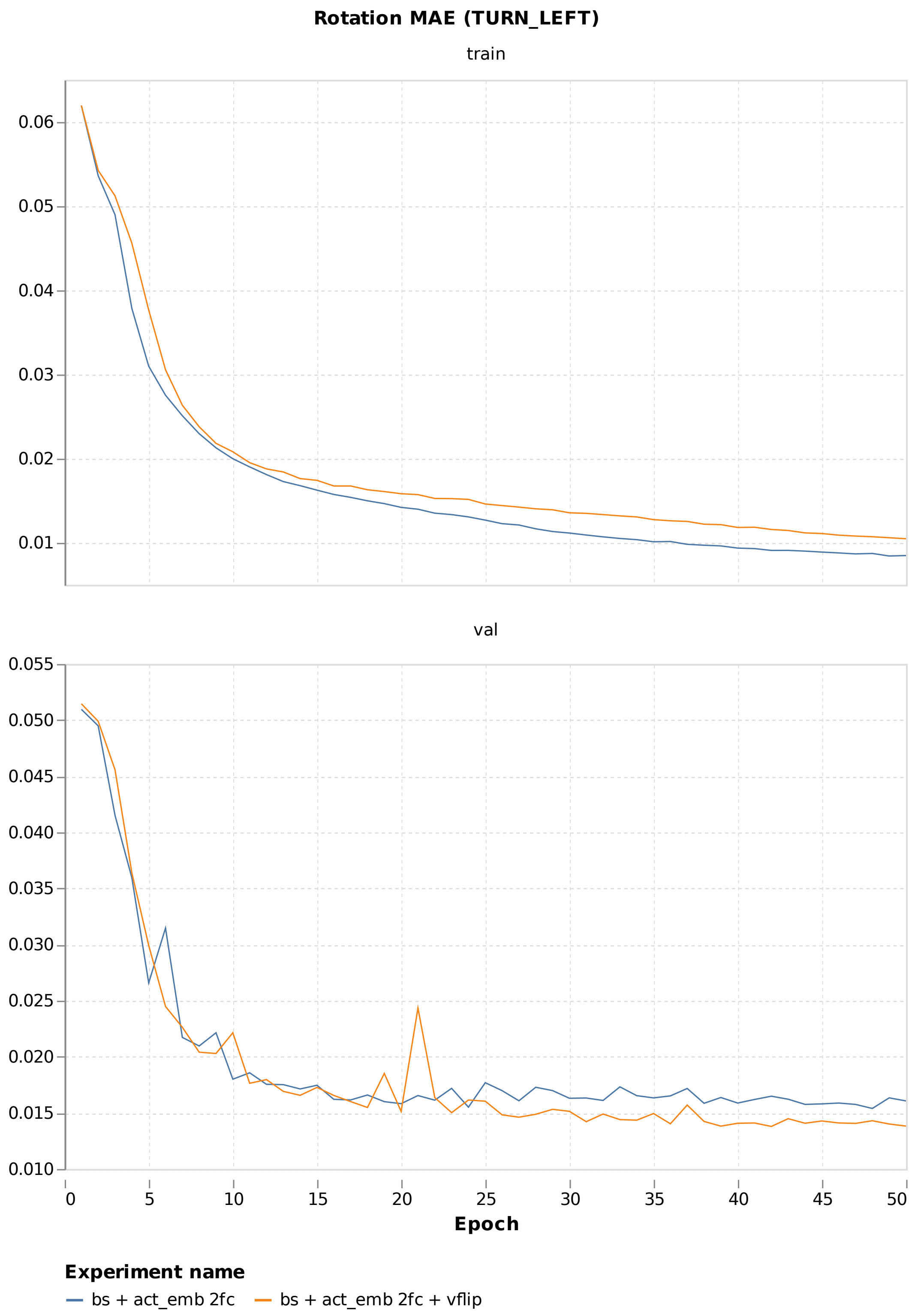


FIGURE A.19: Flip: rotation MAE for TURN_LEFT action.

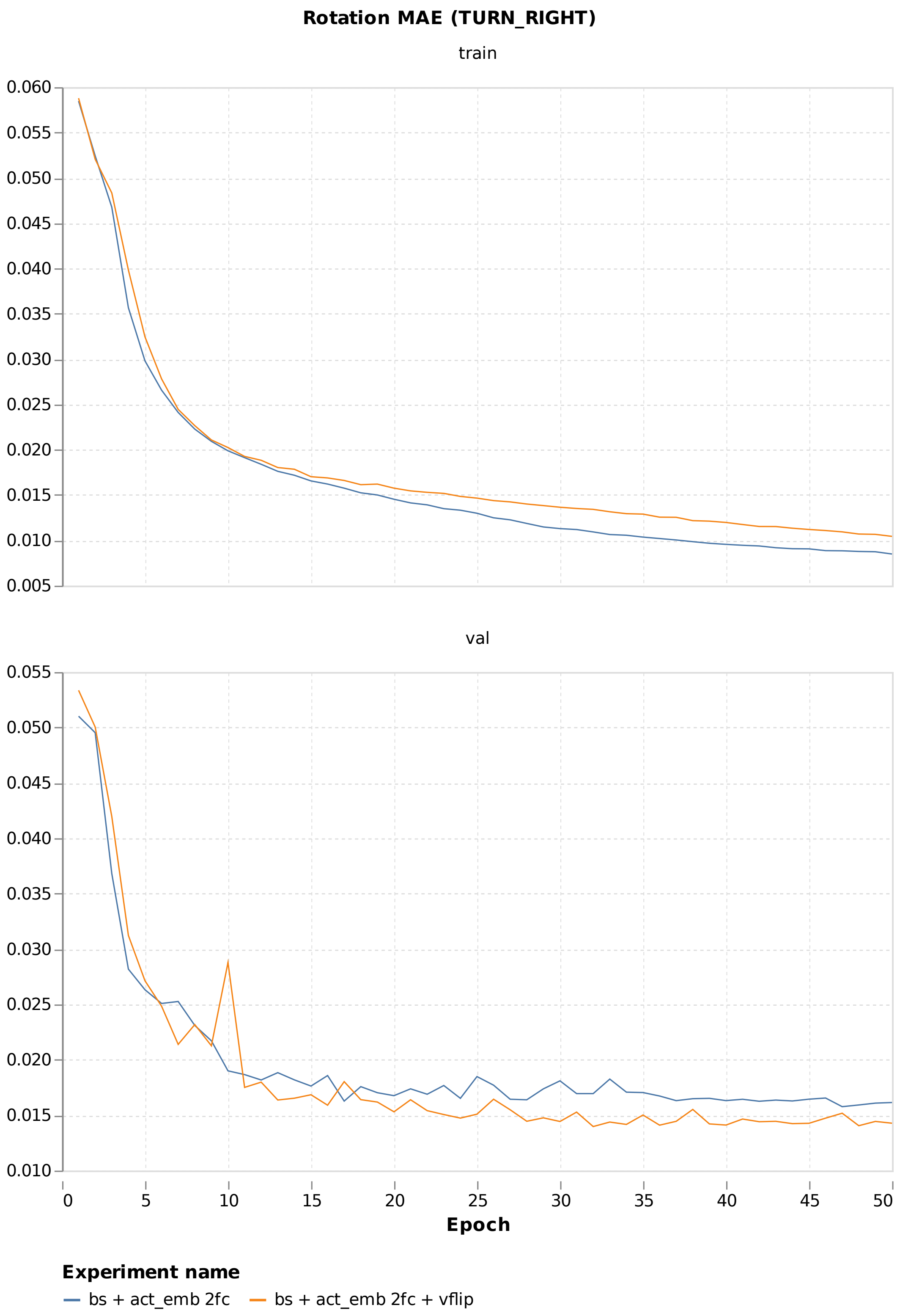


FIGURE A.20: Flip: rotation MAE for TURN_RIGHT action.

A.2.4 Frames Swap

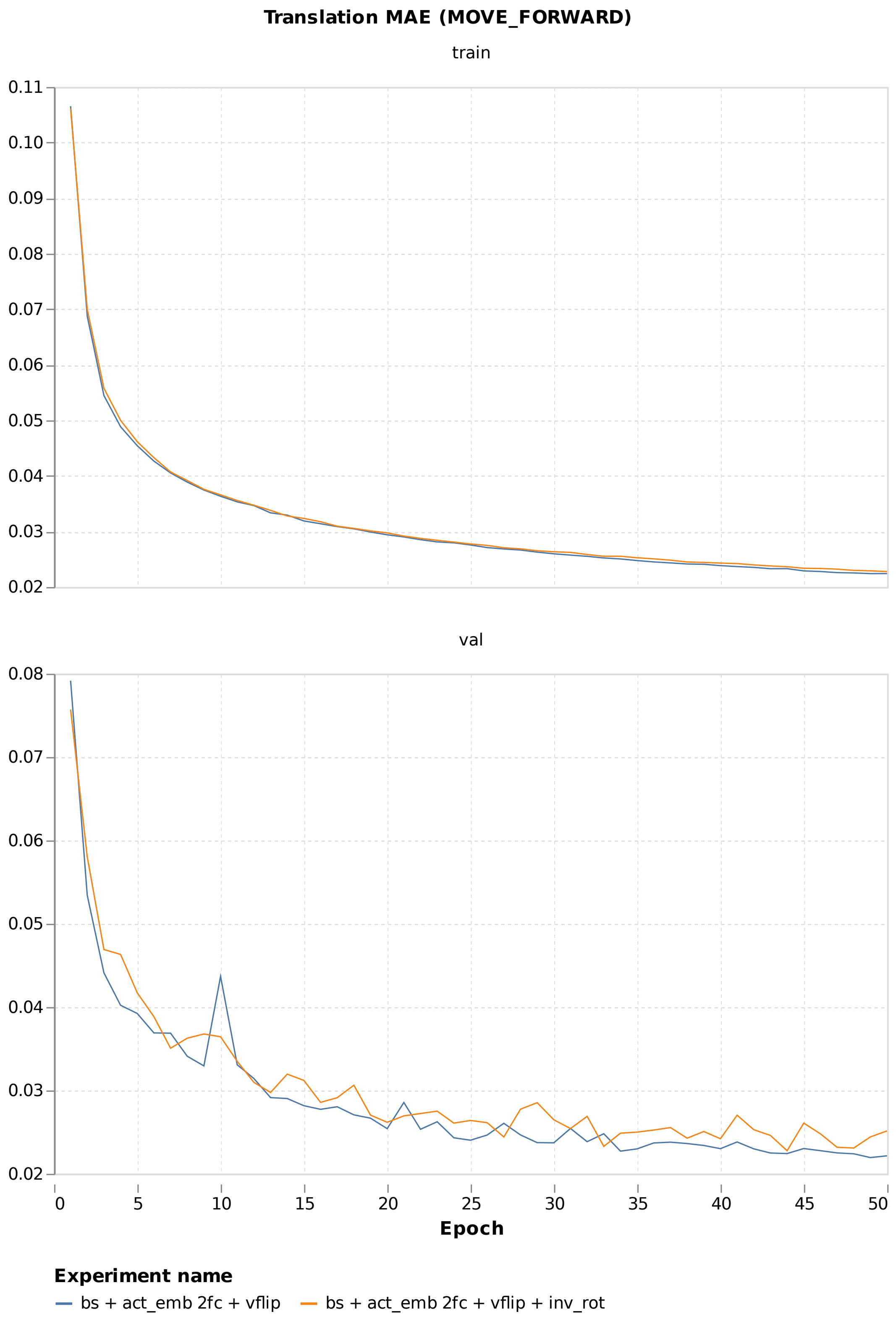


FIGURE A.21: Frames swap: translation MAE for MOVE_FORWARD action.

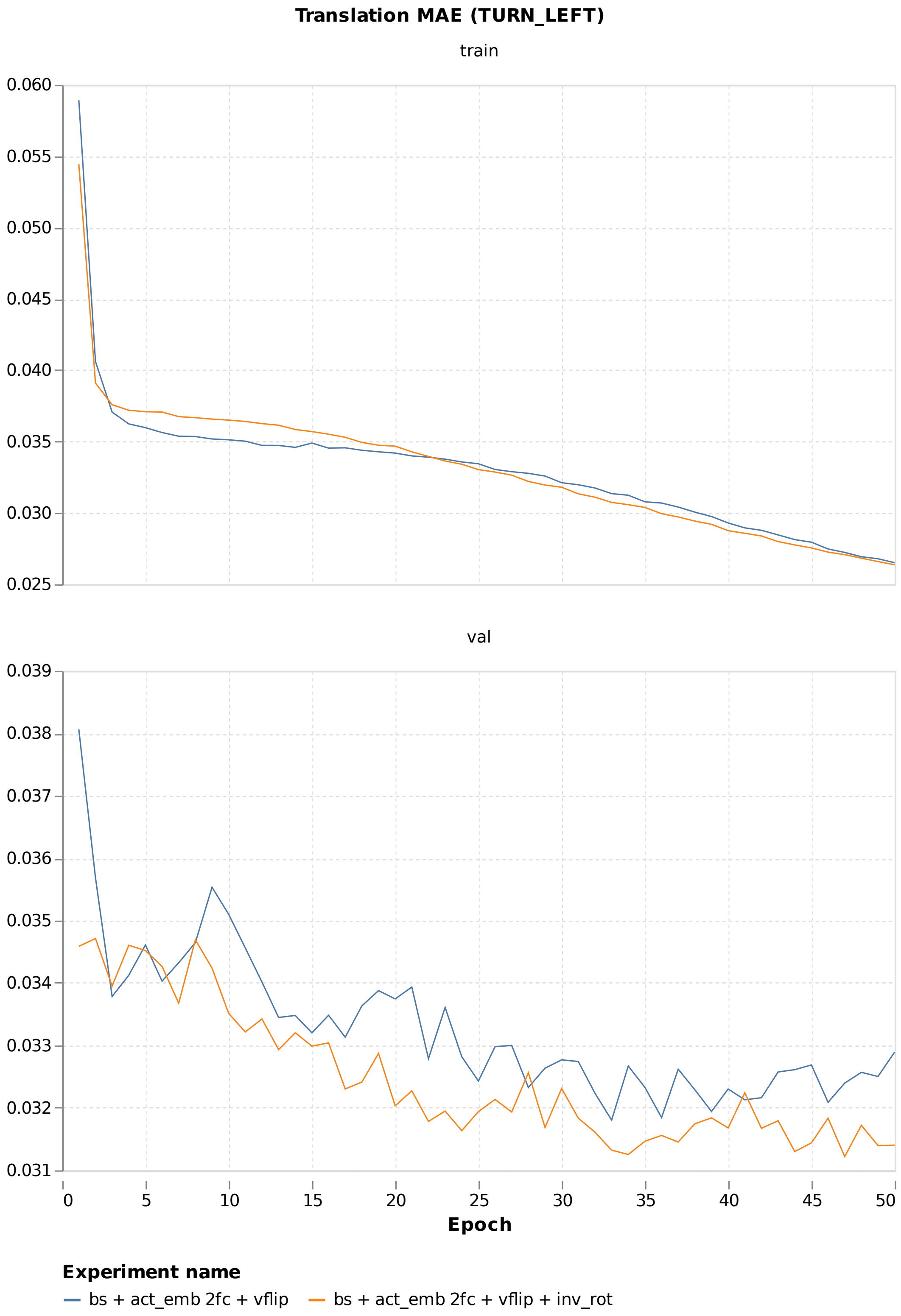


FIGURE A.22: Frames swap: translation MAE for TURN_LEFT action.

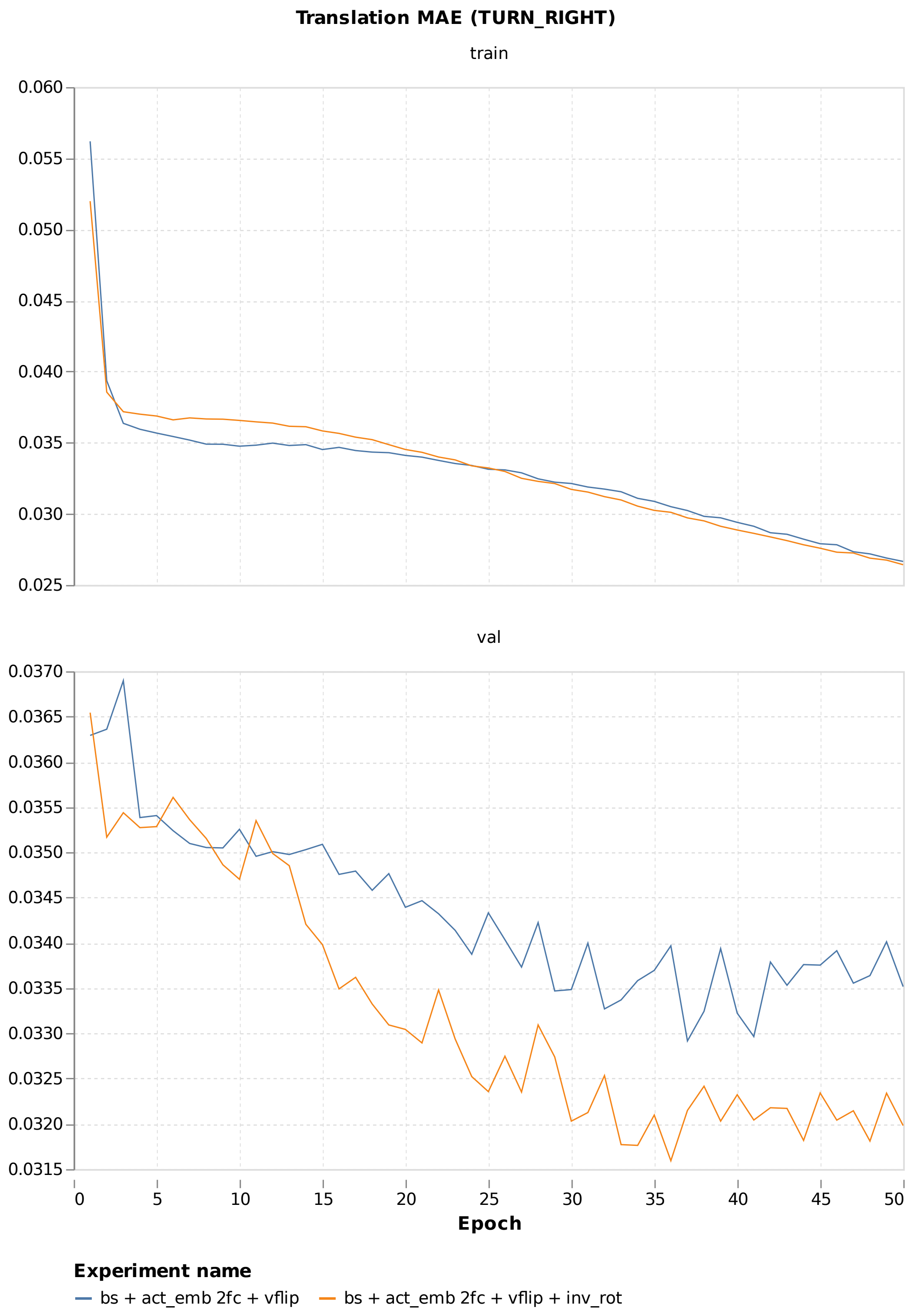


FIGURE A.23: Frames swap: translation MAE for TURN_RIGHT action.

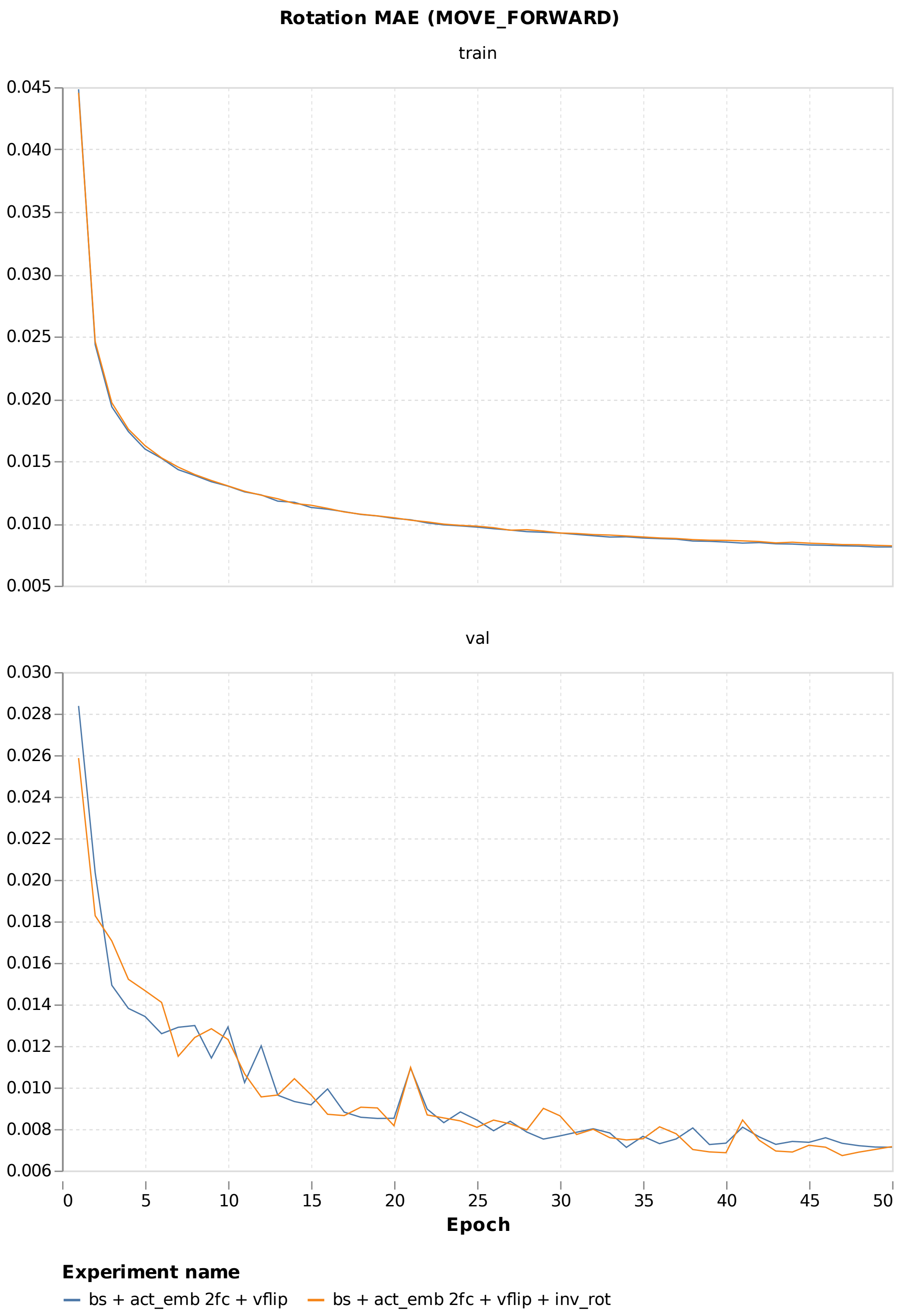


FIGURE A.24: Frames swap: rotation MAE for MOVE_FORWARD action.

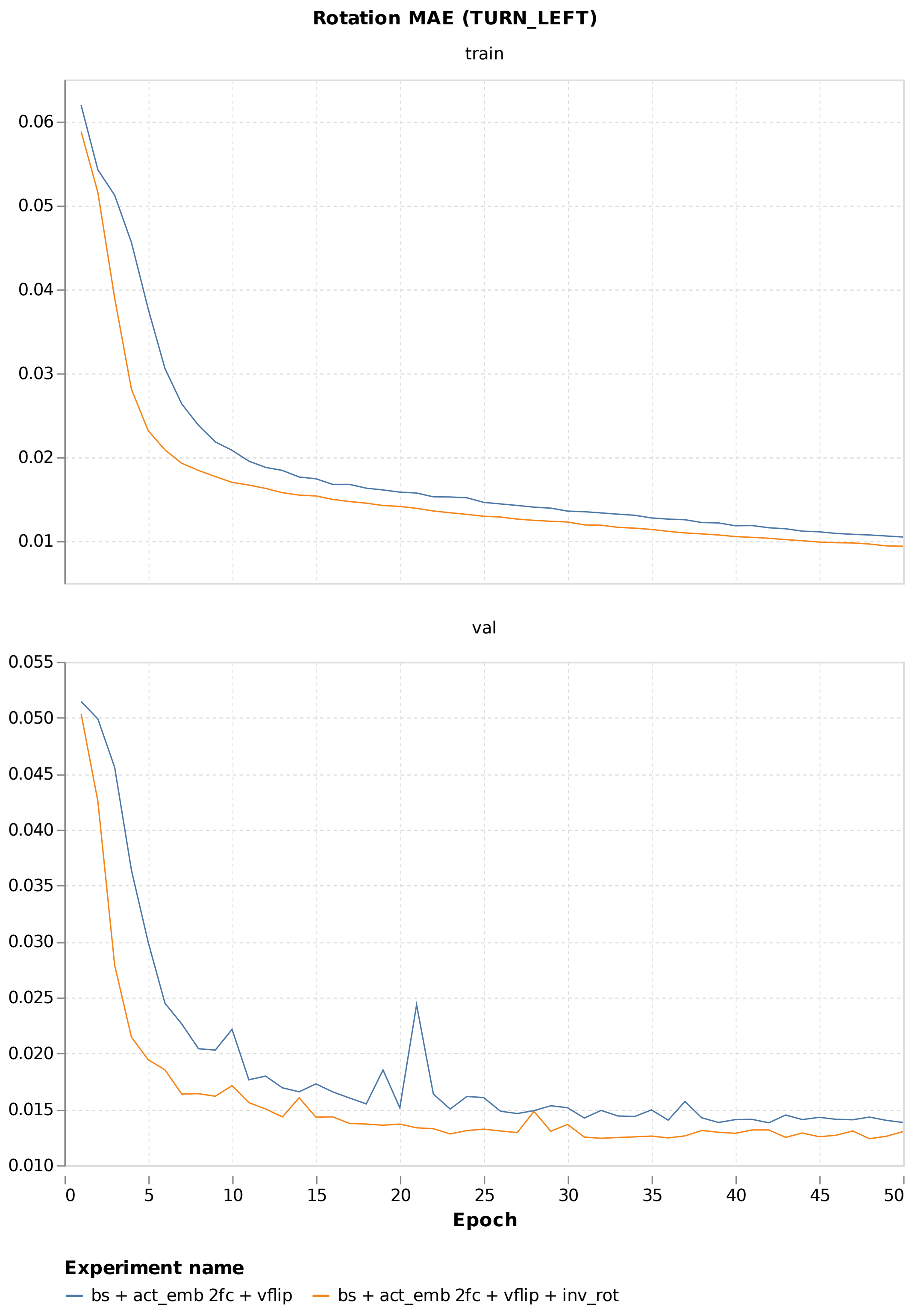


FIGURE A.25: Frames swap: rotation MAE for TURN_LEFT action.

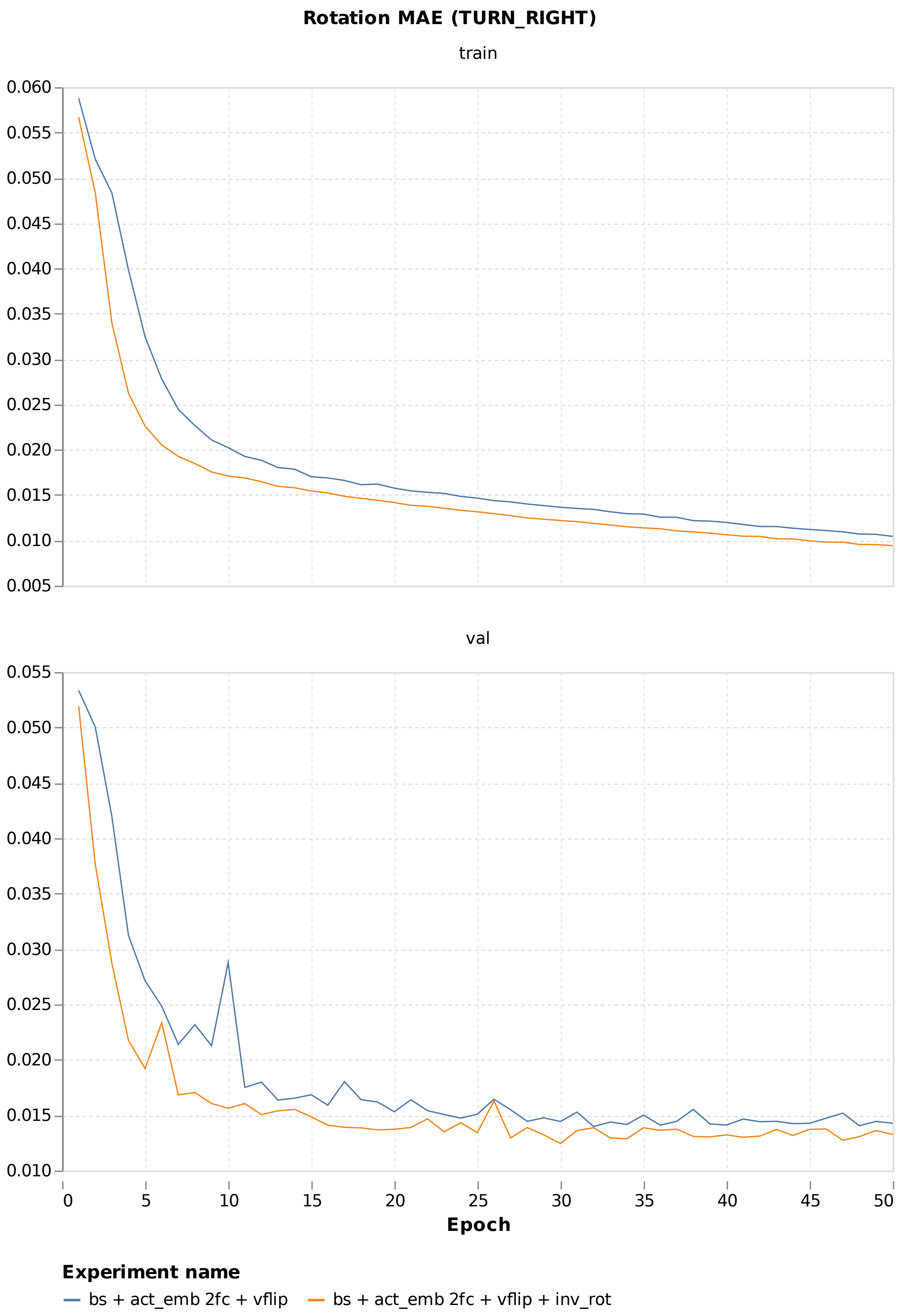


FIGURE A.26: Frames swap: rotation MAE for TURN_RIGHT action.

A.2.5 Separate Model per Each Action Type

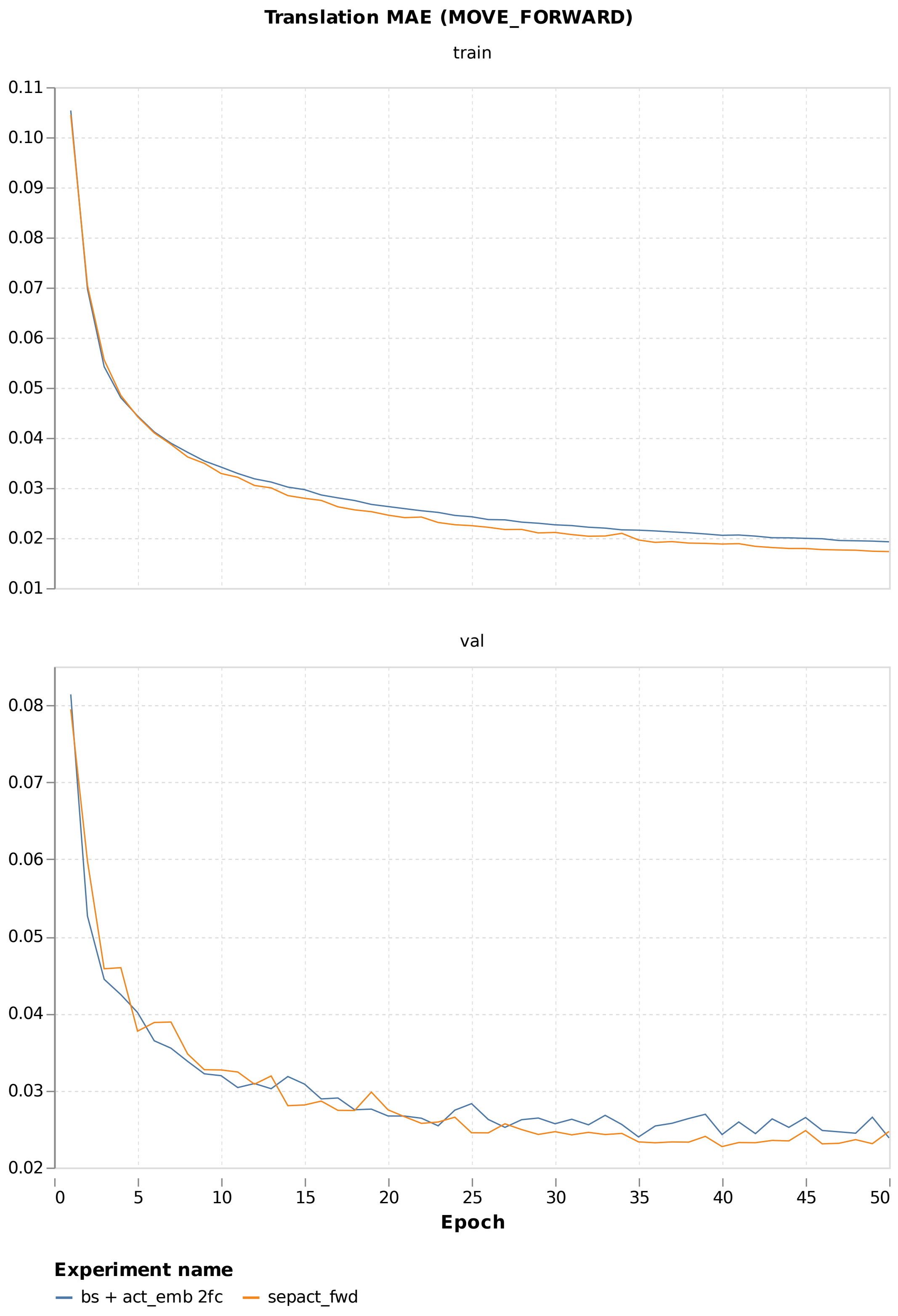


FIGURE A.27: Separate model: translation MAE for MOVE_FORWARD action.

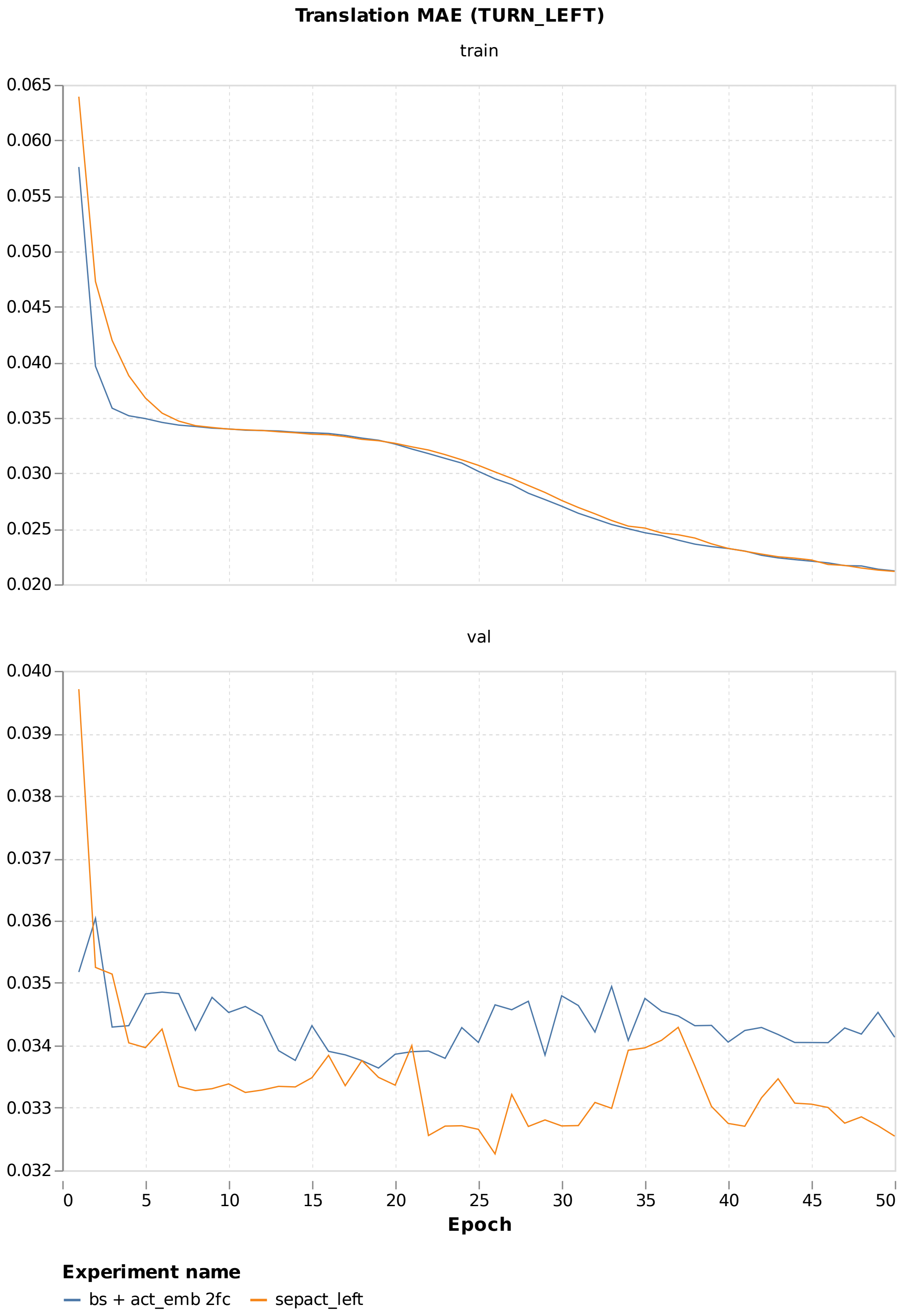


FIGURE A.28: Separate model: translation MAE for TURN_LEFT action.

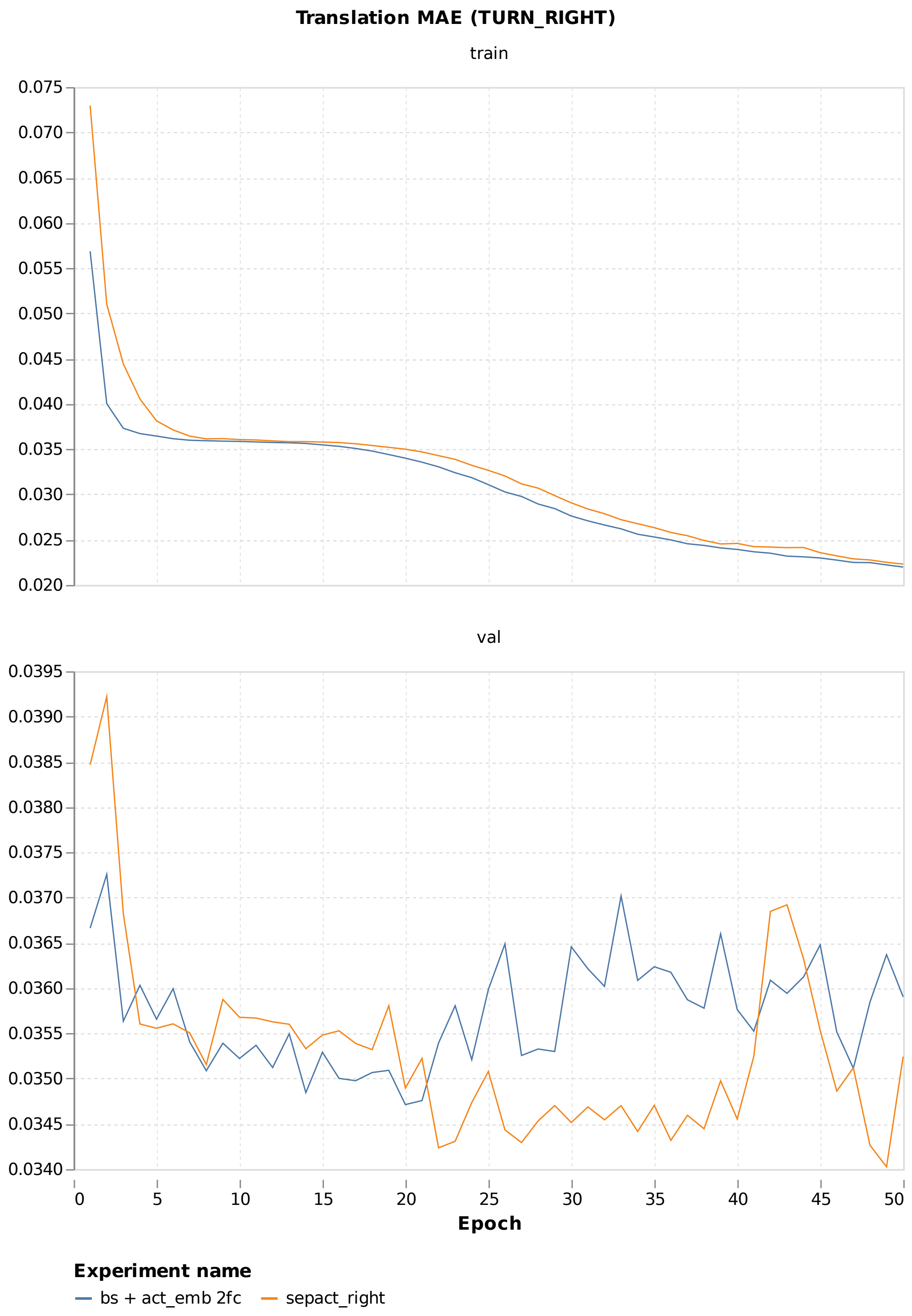


FIGURE A.29: Separate model: translation MAE for TURN_RIGHT action.

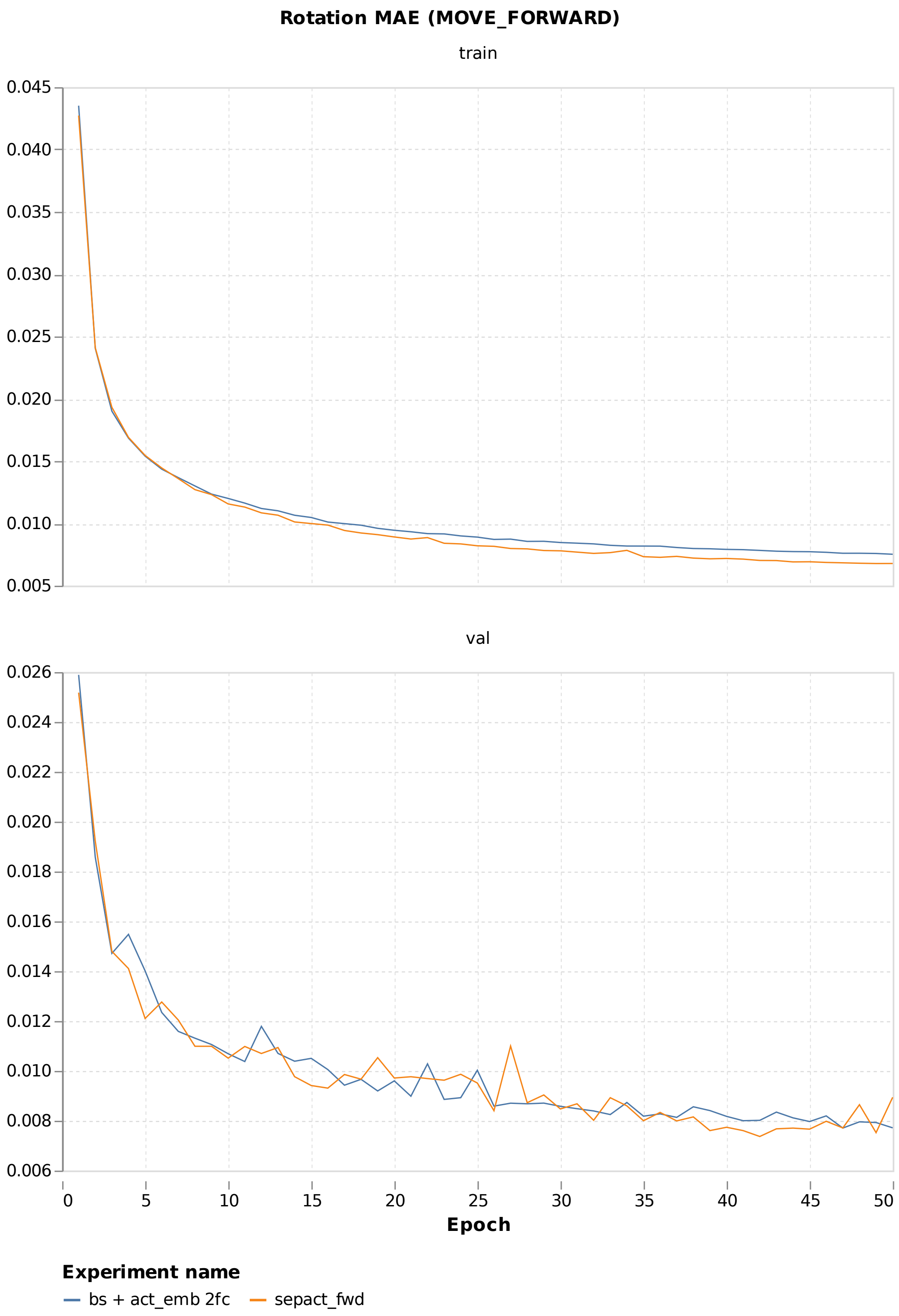


FIGURE A.30: Separate model: rotation MAE for MOVE_FORWARD action.

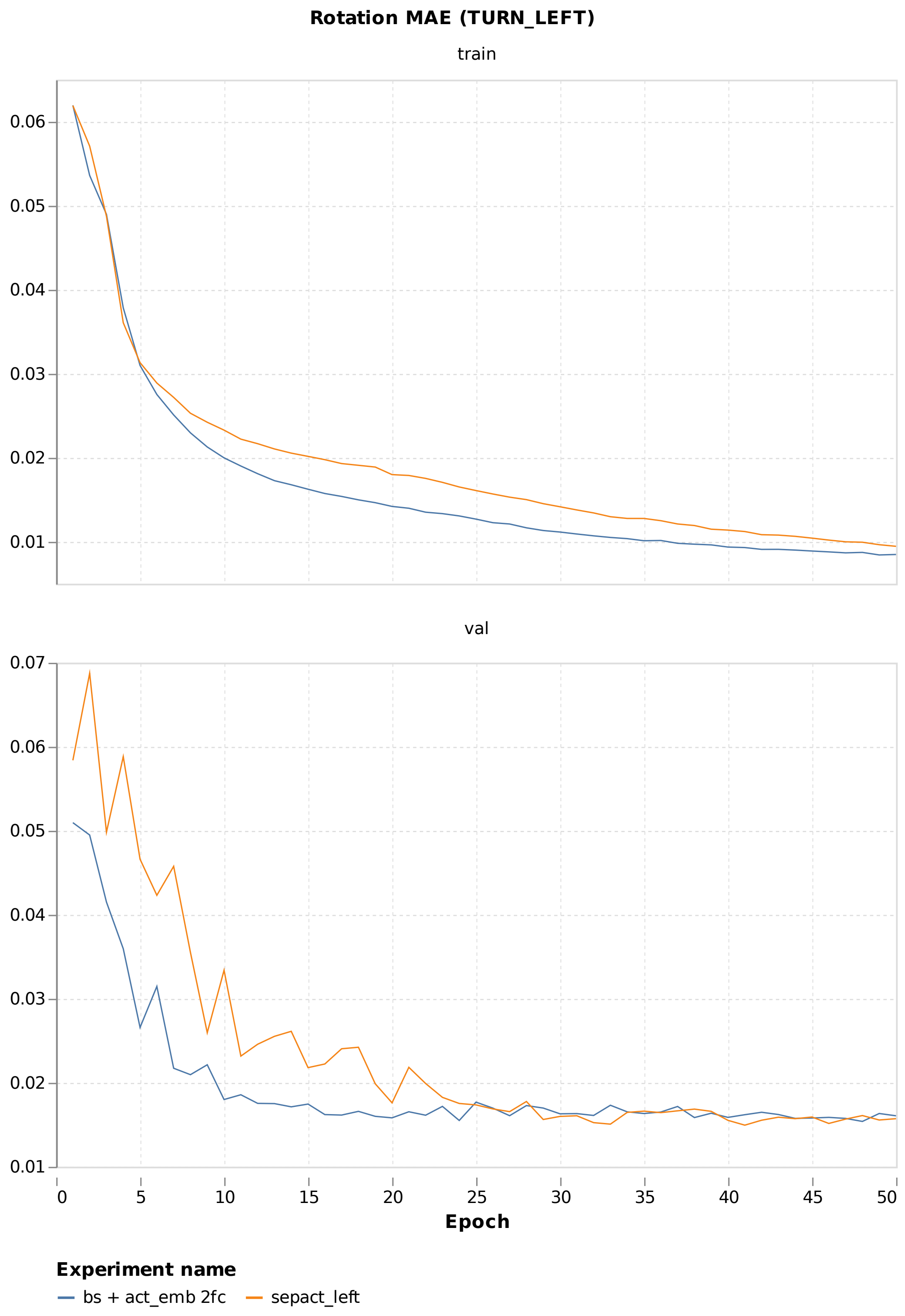


FIGURE A.31: Separate model: rotation MAE for TURN_LEFT action.

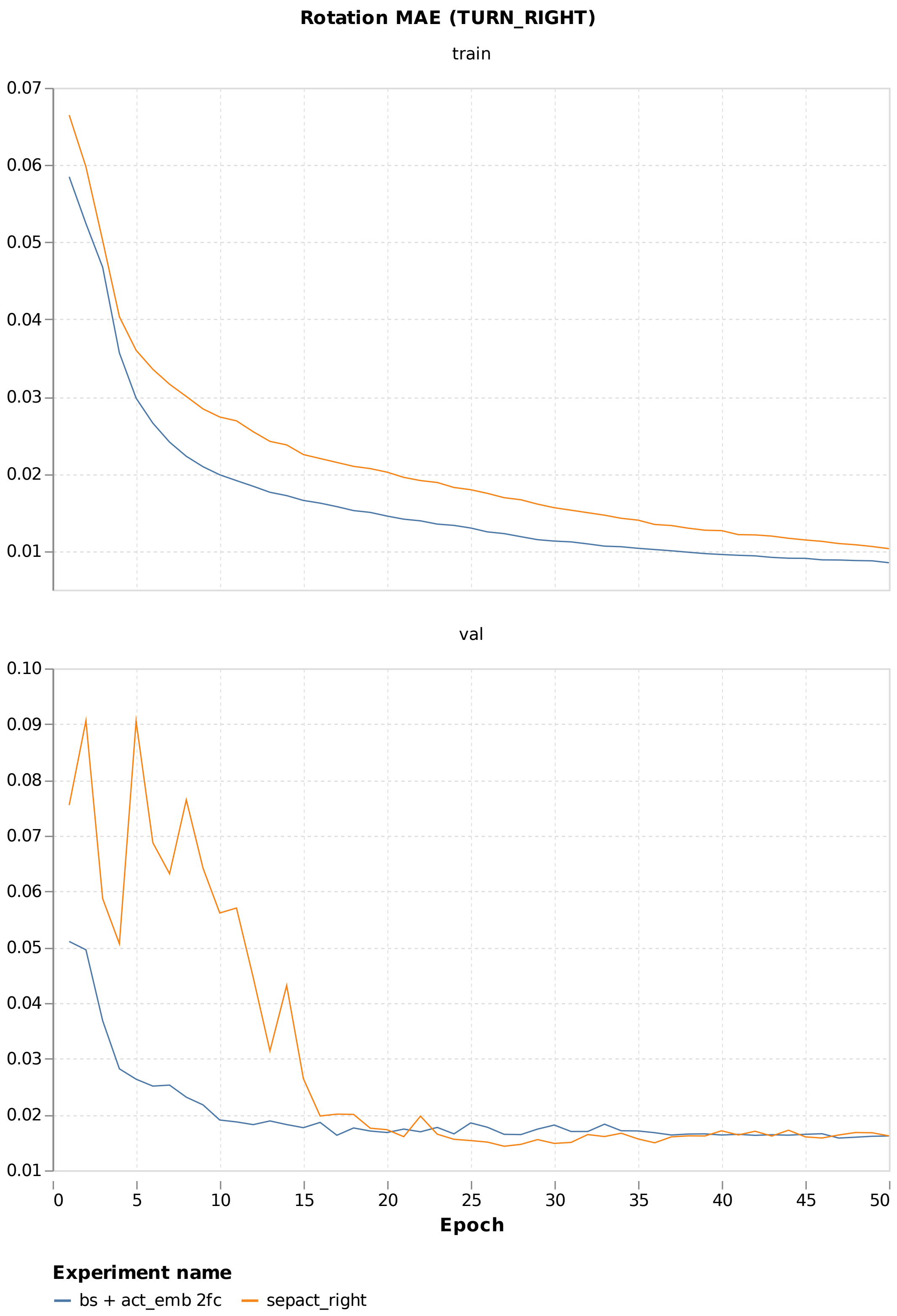


FIGURE A.32: Separate model: rotation MAE for TURN_RIGHT action.

A.2.6 Large Scale Training

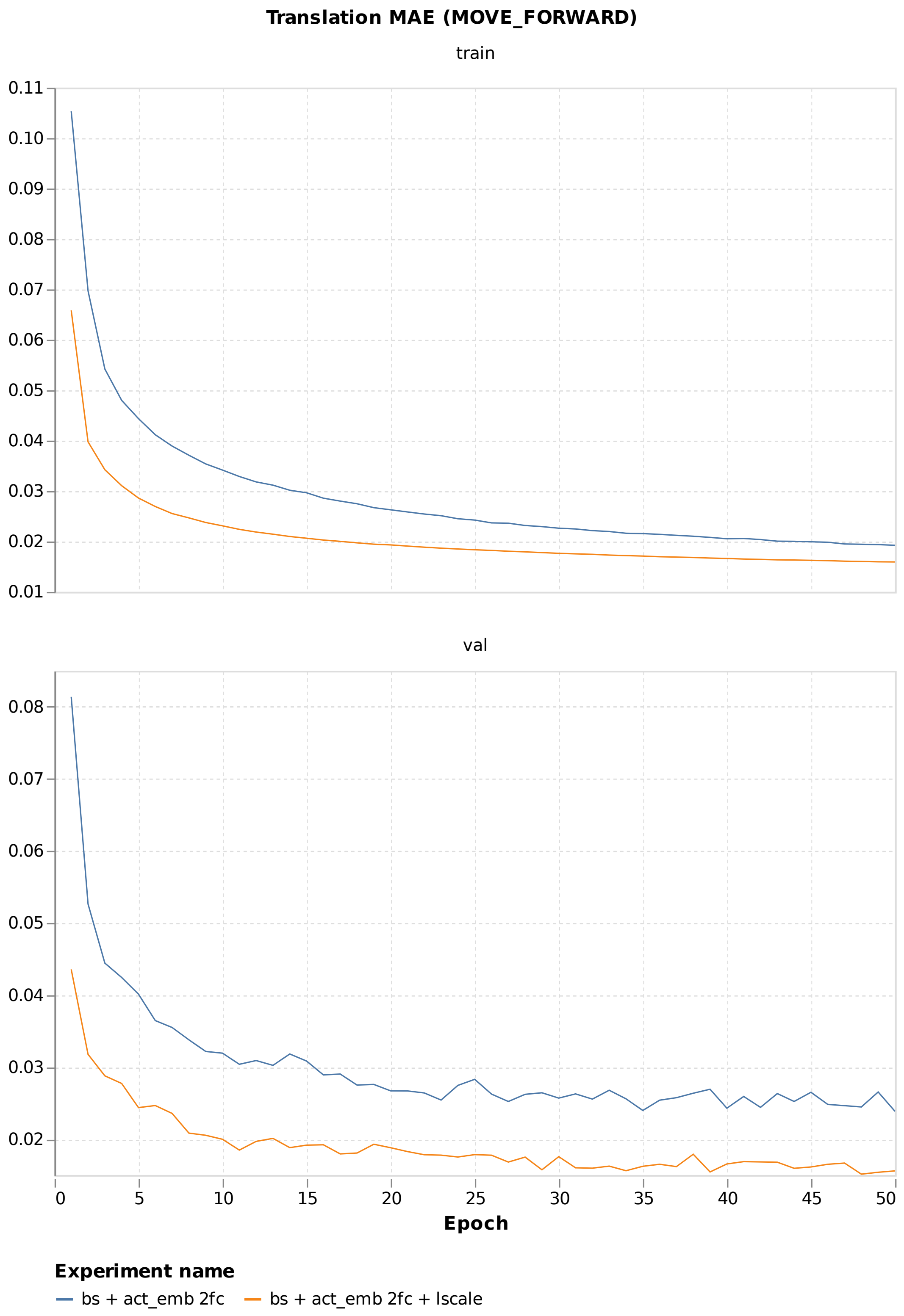


FIGURE A.33: Large scale training: translation MAE for MOVE_FORWARD action.

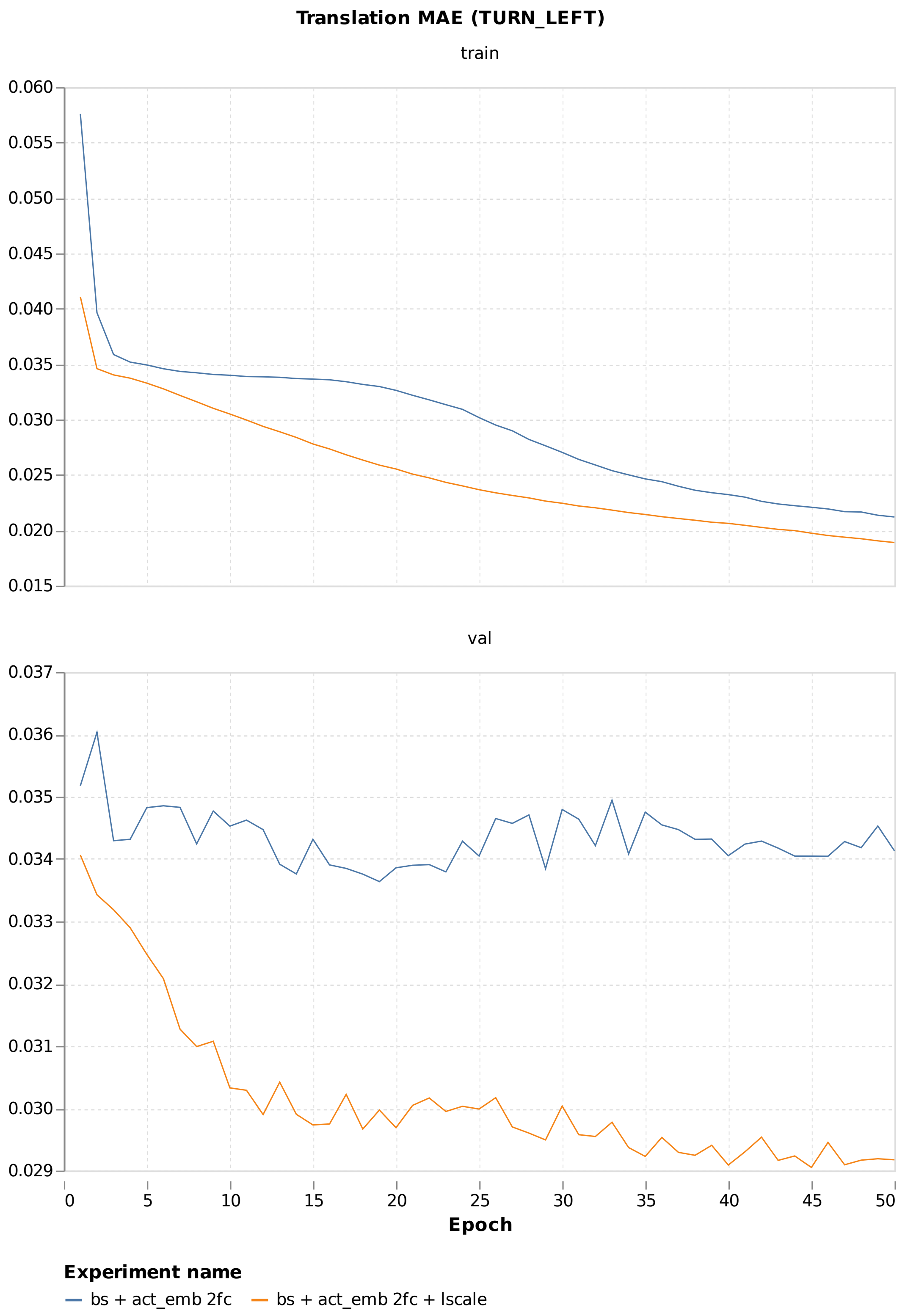


FIGURE A.34: Large scale training: translation MAE for TURN_LEFT action.

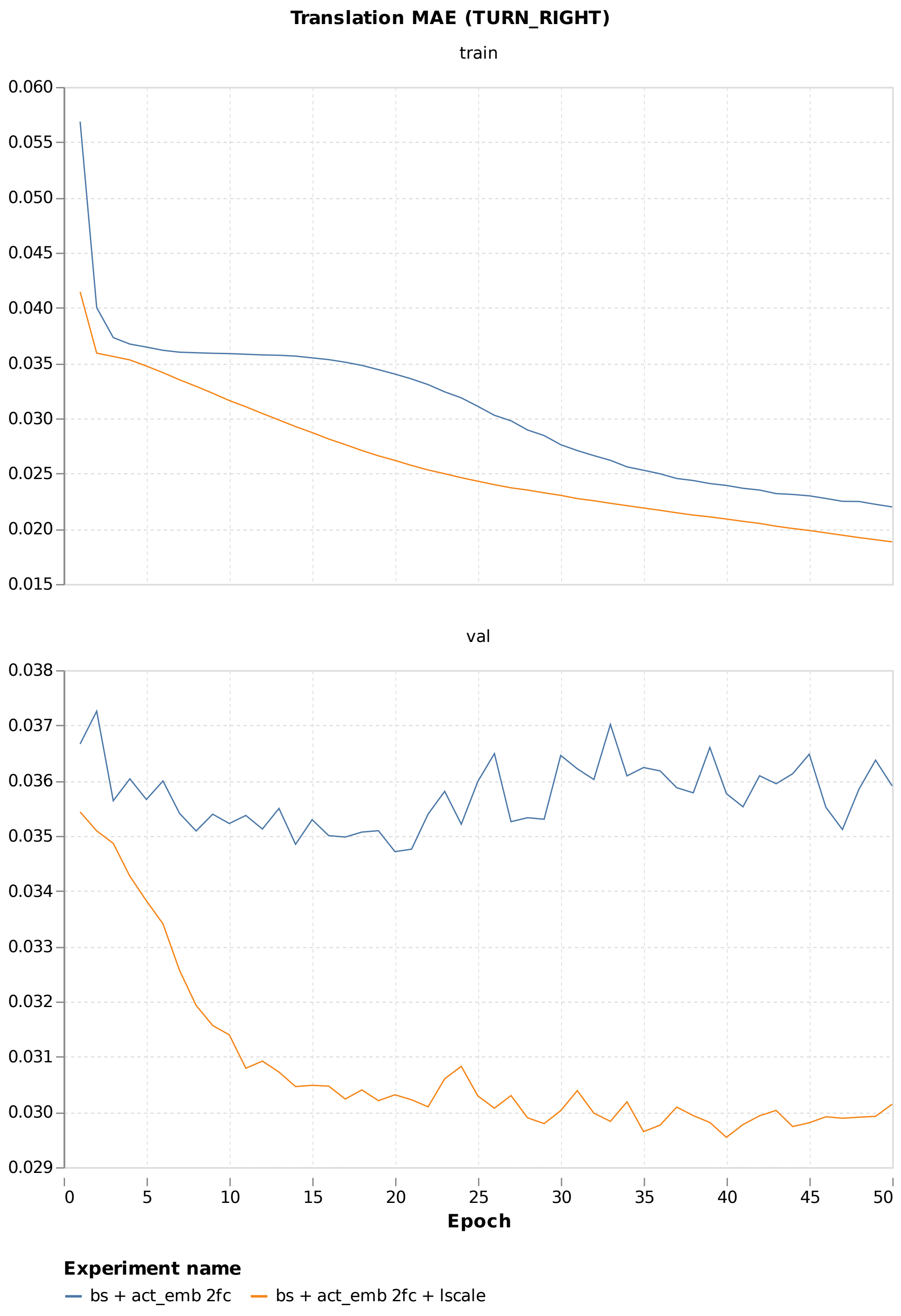


FIGURE A.35: Large scale training: translation MAE for TURN_RIGHT action.

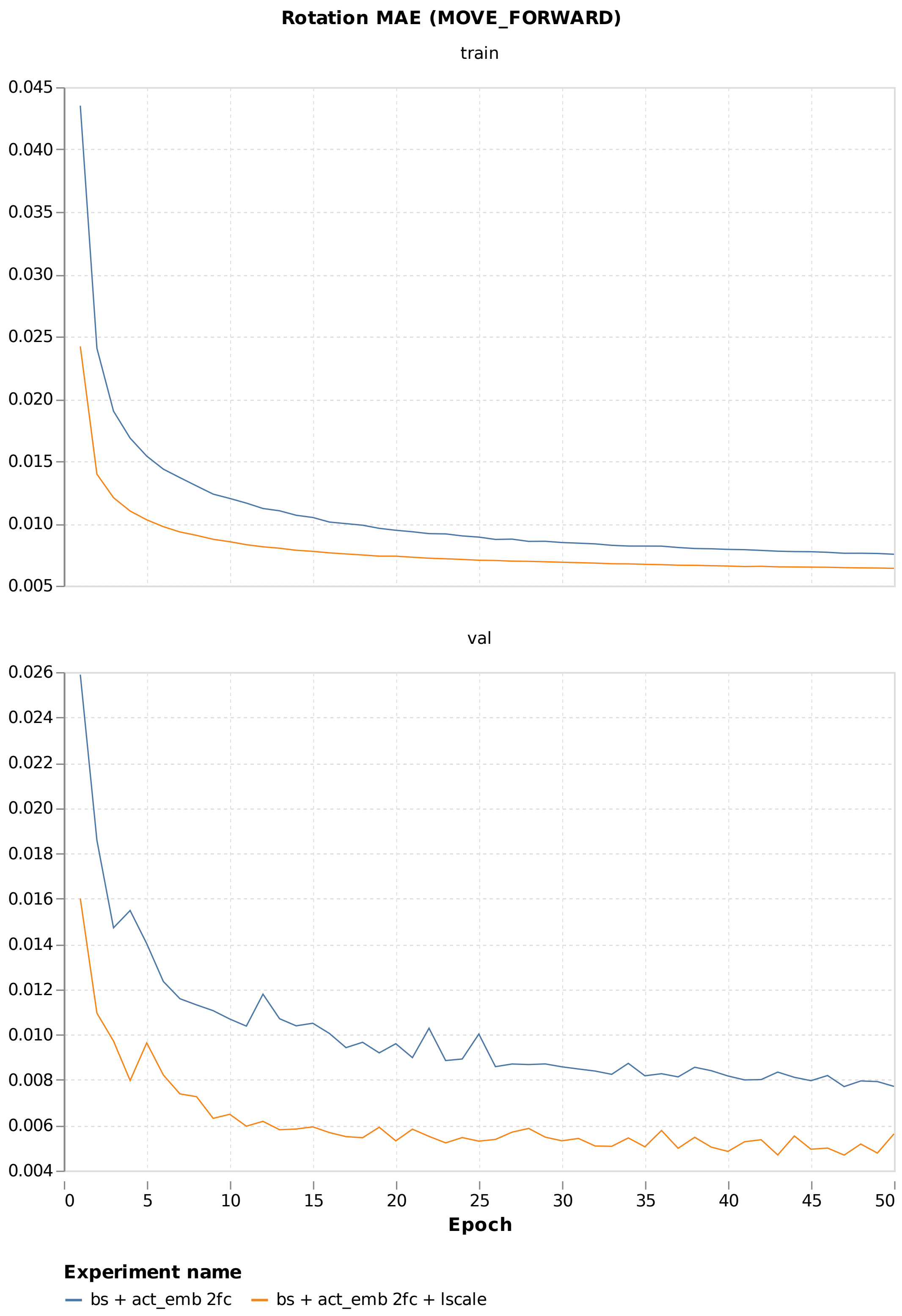


FIGURE A.36: Large scale training: rotation MAE for MOVE_FORWARD action.

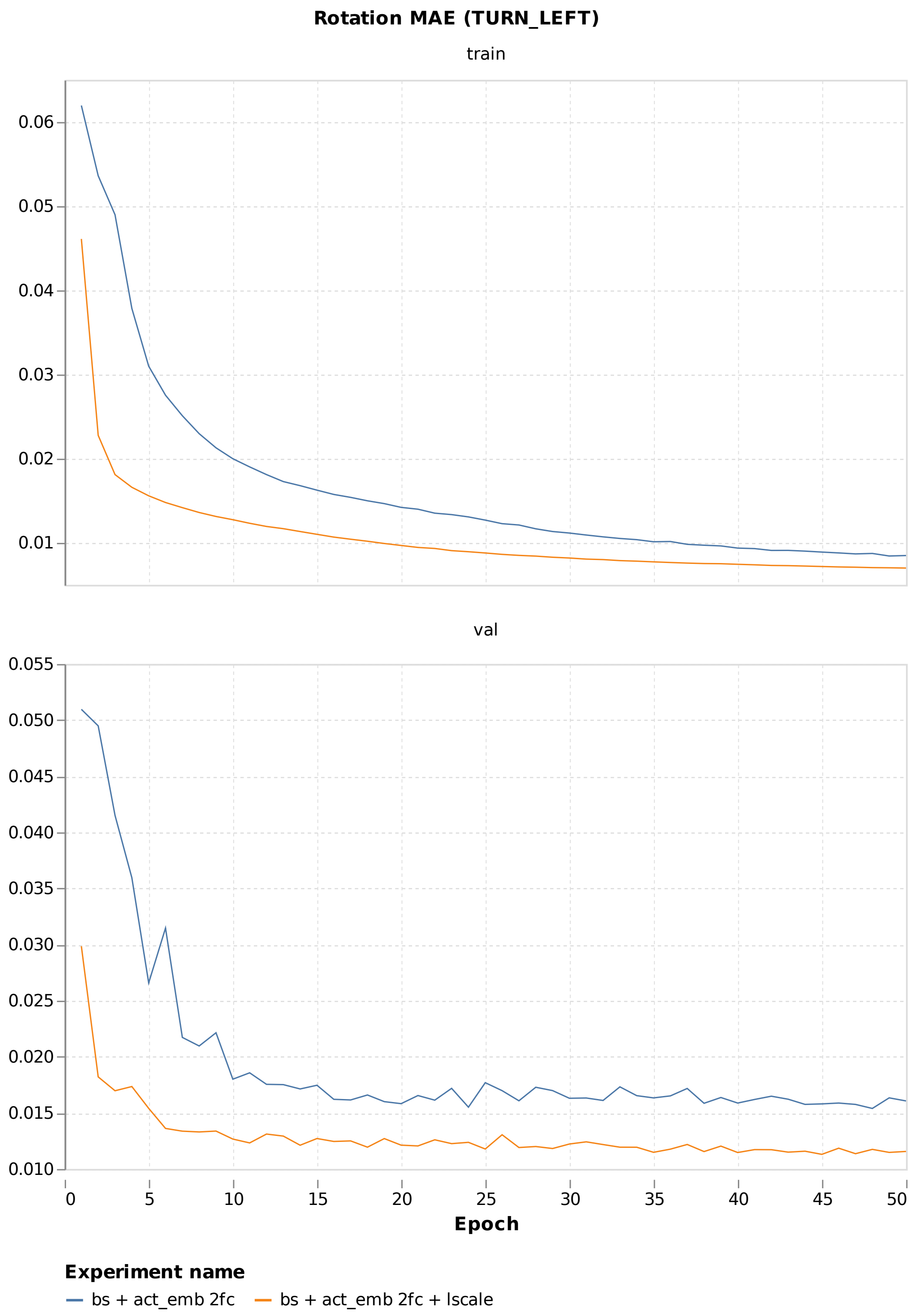


FIGURE A.37: Large scale training: rotation MAE for TURN_LEFT action.

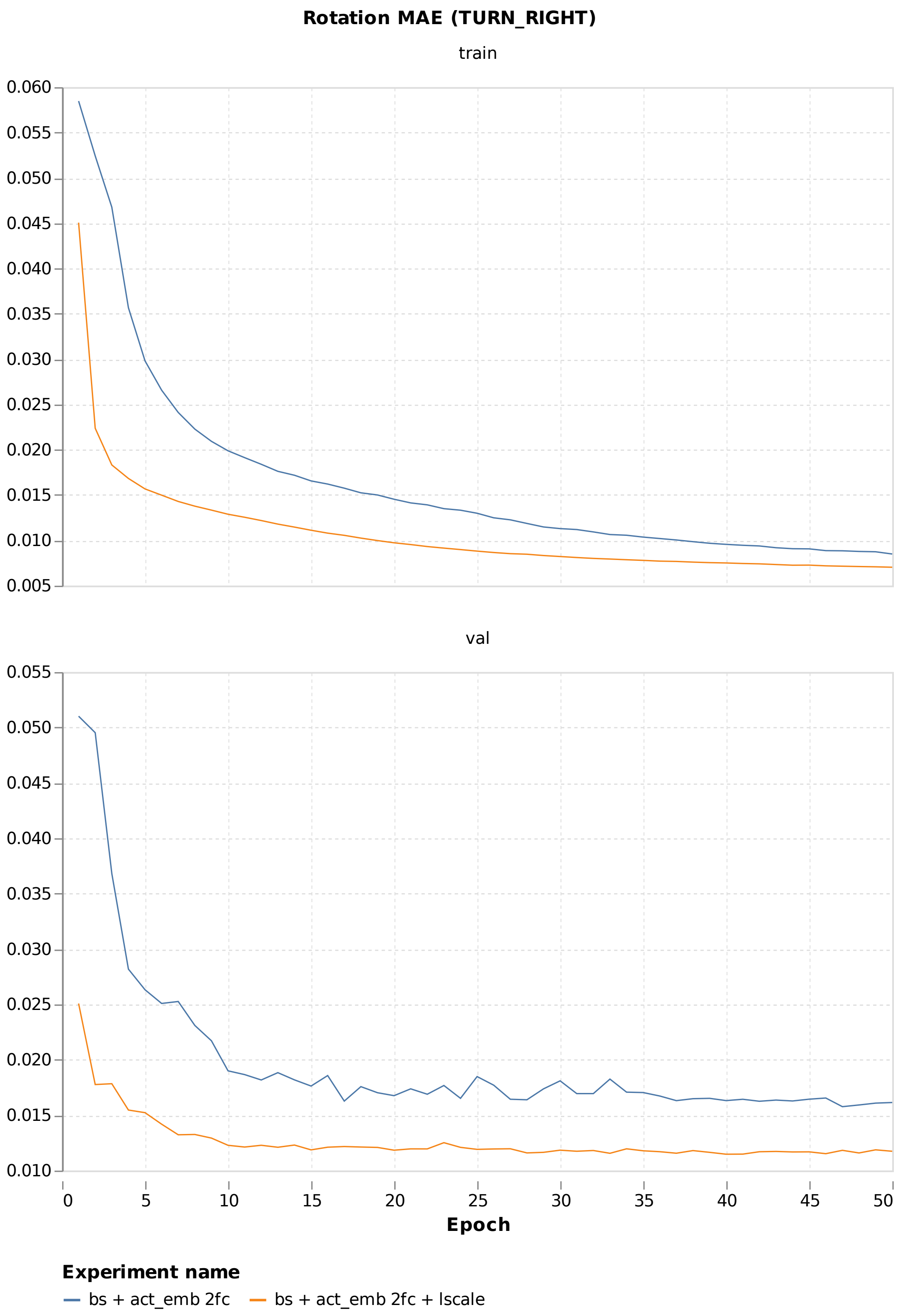


FIGURE A.38: Large scale training: rotation MAE for TURN_RIGHT action.

Bibliography

Anderson, Peter et al. (2018). “On Evaluation of Embodied Navigation Agents”. In: *arXiv preprint arXiv:1807.06757*.

Chang, Angel et al. (2017). “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *2017 International Conference on 3D Vision (3DV)*, pp. 667–676.

Chaplot, Devendra Singh et al. (2020). “Learning to Explore using Active Neural SLAM”. In: *International Conference on Learning Representations*.

Datta, Samyak et al. (2020). “Integrating Egocentric Localization for More Realistic Point-Goal Navigation Agents.” In: *arXiv: Computer Vision and Pattern Recognition*.

Donahue, Jeff et al. (2015). “Long-term recurrent convolutional networks for visual recognition and description”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625–2634.

Durrant-Whyte, H., D. Rye, and E. Nebot (1996). “Localization of Autonomous Guided Vehicles”. In:

He, Kaiming et al. (2016). “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural Computation* 9.8, pp. 1735–1780.

Ioffe, Sergey and Christian Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of The 32nd International Conference on Machine Learning*. Vol. 1, pp. 448–456.

Kadian, Abhishek et al. (2019). “Are We Making Real Progress in Simulated Environments? Measuring the Sim2Real Gap in Embodied Visual Navigation”. In: *arXiv: Computer Vision and Pattern Recognition*.

Kendall, Alex, Matthew Grimes, and Roberto Cipolla (2015). “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2938–2946.

Moravec, Hans (1984). “Locomotion, Vision and Intelligence”. In: *Proceedings of Robotics Research - The First International Symposium*. Ed. by Michael Brady and Richard Paul. MIT Press, pp. 215–224.

Ramakrishnan, Santhosh K., Ziad Al-Halah, and Kristen Grauman (2020). “Occupancy Anticipation for Efficient Exploration and Navigation”. In: *ECCV (5)*, pp. 400–418.

Savva, Manolis et al. (2017). “MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments”. In: *arXiv preprint arXiv:1712.03931*.

Savva, Manolis et al. (2019). “Habitat: A Platform for Embodied AI Research”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9339–9347.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Wang, Sen et al. (2017). “DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050.

Wijmans, Erik et al. (2020). “DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames”. In: *Eighth International Conference on Learning Representations*.

Wu, Yuxin and Kaiming He (2018). “Group Normalization”. In: *arXiv: Computer Vision and Pattern Recognition*.

Xia, Fei et al. (2018). “Gibson Env: Real-World Perception for Embodied Agents”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9068–9079.