# Success Classification for Object Navigation

by

Albert Yue

B.S., Computer Science and Engineering and Mathematics,
Massachusetts Institute of Technology (2021)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022



Author .....
Department of Electrical Engineering and Computer Science
May 13, 2022

Certified by .....
Pulkit Agrawal
Assistant Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

**Success Classification for Object Navigation**

by

Albert Yue

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Object navigation is the embodied task of navigating to an instance of a specified object in unseen environments. Previous work has made impressive progress on the problem, but there remains much room for improvement with current state-of-the-art methods reaching a success rate of less than one in three. In this work, we evaluate a state-of-the-art approach, identifying false positives in object detection as the main point of failure. We propose introducing a new module to verify success when the agent attempts to stop. We introduce a learning-based classifier that learns and compares embeddings for visual observations and object categories and find that it works well at predicting success, outperforming both naive baselines and a heuristic-based classifier. We also find no improvement when using a ensemble model for semantic segmentation, although we believe there is more to be tested before arriving at a conclusive judgement.

Thesis Supervisor: Pulkit Agrawal
Title: Assistant Professor

# Acknowledgments

I would like to thank Prof. Pulkit Agrawal for his mentorship and guidance on this project.

Thank you to Haokuan Luo, Felix Wang, and Zhang-Wei Hong for their collaboration on object navigation work, as well as to all the other members in the lab that helped me with my compute, engineering, and research questions.

Thank you to my friends and family for all of their support in the past year, and for every point that has led up to now. Thank you especially to Aaditya Singh for all of the fruitful conversations about research and philosophy.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the advent of household robots, autonomous navigation has become a necessary skill for embodied, artificial agents to perform. Consider the scenario where a robot is brought to a new unseen environment and asked to find a target object, such as a sink or table. For the robot to succeed in this task, known as object navigation, it must not only be able to view the environment and detect the target object, but also effectively explore the unseen environment and successfully navigate to the object once found.

To this end, the Habitat ObjectNav Challenge [3] has been held for the past three years, providing a common dataset and standardized metrics to compare proposed techniques. The challenge uses Matterport 3D [4], a set of high-quality scene meshes of indoor spaces, as environments for navigation. Contestants additionally have access to a dataset of episodes, which include a specified scene, target object, starting state, and valid locations for success. Agents can train using the train and validation scenes and episode dataset, and are evaluated on a held-out test set of unseen scenes. The current state-of-the-art (SOTA) approach submitted to the challenge uses reinforcement learning to train an end-to-end policy network [24]. While these techniques can succeed for a diverse set of homes and target objects, they are still far from perfect, with the top performance have a success rate of less than one in three.

In this work, we evaluate an existing state-of-the-art method in object navigation, identify the current challenges and major failure modes of the approach, and work to

address them. We classify evaluation episodes using a set of failure categories based on previous analysis [24], extending the analysis in that work, and identify object detection and semantic segmentation as the primary bottleneck of current agents. We propose introducing a success verifier module to the agent to mitigate failures in object detection, and investigate different methods to construct and train such a classifier. We find that a learned success classifier can outperform a heuristic baseline and present the following results with regards to such a classifier: 1) learning and comparing representations of the visual observations and target object performs better than training a network that takes these as inputs, 2) transferring trained models from ground-truth to predicted semantic inputs performs well initially but is comparable at convergence, and 3) different methods for converting semantic predictions into inputs for the downstream classifier perform similarly. We also present results using a ensemble model for semantic segmentation via multiple independent final layers, and find that the model performs similarly to the original semantic segmentation model.

# Chapter 2

# Background and Related Work

## 2.1 Object Navigation

In object navigation [2], an agent is placed in an unseen environment and must navigate to an instance of the specified object category. The agent is given no information about the environment beforehand (e.g. a map) and must take actions to observe the environment. It has access to several onboard egocentric sensors: RGB-D cameras and a GPS+Compass that gives a global position and orientation. The agent has a discrete action space: *move forward*, *turn left*, *turn right*, and *stop*. The agent may have two additional actions to provide a second axis to rotate the camera, namely *tilt up* and *tilt down*. An episode is considered successful if the agent stops within 1 meter of an object of the target category and can see the object from some camera orientation within 500 timesteps (corresponding to 500 actions). We use two metrics to evaluate the performance of agents: success rate, the percentage of episodes in which the agent succeeds; and success weighted by path length (SPL), defined as

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

where $N$ is the number of episodes, $S_i$ is a binary indicator of success, $l_i$ is the shortest path length to an object of the specified category, and $p_i$ is the path length taken by the agent. SPL provides a measure of path efficiency along with success, as more

efficient agents will score higher in terms of SPL. Note that a perfect score of 1 on SPL is infeasible in object navigation, as the agent has no information about the environment beforehand and must spend time exploring the environment first.

For our work, we use the Habitat simulator [19] to simulate photo-realistic indoor environments and the Habitat Challenge framework [3]. In this framework, the forward step size is 0.25 meters, and turn and tilt angles are 30 degrees. We use the Matterport3D (MP3D) [4] dataset, a set of high-quality scene meshes of indoor spaces that can be used as 3D environments for simulation. MP3D contains 90 scenes across the train/val/test splits and 40 labelled semantic categories, although we only use the 21 categories hand-picked by the Habitat Challenge [3] as target categories.

## 2.2 Existing work in Object Navigation

Object navigation has been part of the 2020 and 2021 Habitat Challenges [3], and prior works have found some success in the task. It should be noted that a simple RGB-D CNN+RNN baseline using a ResNet-50 [10] visual encoder and DD-PPO [23] gets a 0.00 success rate on the test set [1], despite its near-perfect performance in point navigation, where the agent navigates to a specified relative coordinate [23]. More recently, Ye et al. (2021) won the 2021 contest by building on top of the baseline, adding semantic features, auxiliary tasks and an exploration reward (see Figure 2-1) [24]. Similarly, Khandelwal et al. (2021) replace the visual encoder with a pretrained CLIP ResNet-50 [12]. Maksymets et al. (2021) also builds off the CNN+RNN baseline, but instead augments the training scene dataset by inserting objects to increase object and trajectory diversity [17].

Along with end-to-end learning approaches, other approaches maintain an intermediate state representation to aid in navigation. Commonly, researchers use simultaneous localization and mapping (SLAM) to construct a map to use for planning [7, 21]. Chaplot et al. (2020) maintain a semantic map and use it to predict likely next locations to explore and find the target object [6]. This approach was further

[1] On the leaderboard https://eval.ai/web/challenges/challenge-page/802/leaderboard/2192

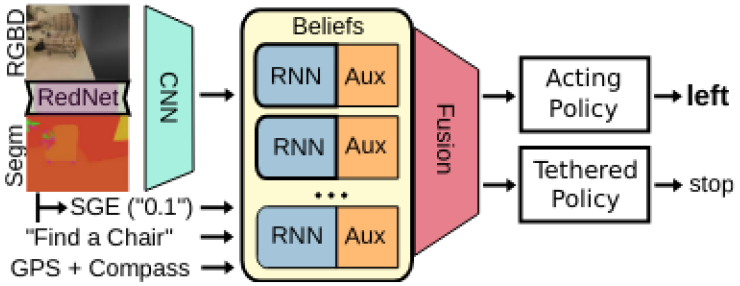



Figure 2-1: The model architecture of the 2021 Habitat ObjectNav Challenge winner, taken from their paper [24], which we use as the basis for our own agent.

iterated on in [5], in which they introduce an intrinsic reward to observe objects from angles that provide high semantic confidence and use collected trajectories to further finetune the semantic segmentation model. Luo et al. (2022) present a similar agent for object navigation, using a simple semantic-agnostic exploration strategy, object verification, and a multi-scale collision map [16].

## 2.3 Semantic Segmentation

Scene understanding is one of the biggest goals in computer vision, and semantic segmentation comprises a large part of this field. Recent work in object navigation has used pretrained semantic segmentation networks [10, 9, 11] to provide their policies with semantic information.

In embodied tasks, agents make observations over time, in which observations between time steps are highly correlated with each other as the agent remains in the same environment. For example, if the agent sees a table and moves towards it, the next observations will most likely contain the same table until the agent moves past or turns away from it. However, classic semantic segmentation models make predictions on a per-frame basis and do not store information about past frames or predictions. Therefore, they can be prone to making inconsistent predictions, as seen in Figure 2-2. Some prior works have tried to improve accuracy by exploiting these temporal relationships, such as [18] and [26], which use optical flow networks to propagate predictions. As optical flow is computationally intensive, and thus impractical


(a) $t = 0$


(b) $t = 1$

Figure 2-2: RGB, depth, and predicted semantics for two consecutive frames in a bathroom. The sink is correctly labelled at time $t = 0$ but then labelled as multiple object categories (sink, table, cabinet) at $t = 1$.

for real-time inference, more recent approaches instead incorporate temporal consistency as a component of the loss and via knowledge distillation during training [15]. Similarly, Wang et al. (2021) construct a memory and use attention to relate the current frame and previous frames [22]. With regards to object navigation, Luo et al. (2022) introduce a learned model that incorporates past semantic information to verify positive target detection and mark false positives [16].

# Chapter 3

# Failure Mode Analysis of SOTA

To guide our work, we investigate the current performance of the state-of-the-art in object navigation. Namely, we focus on the 2021 Habitat Challenge winner, which we refer to as the End-to-End Auxiliary Task agent (EEAux) [24]. We group failures into the following modes, which can then be broadly grouped as failures related to target object detection, exploration, policy failures after finding the object, and other miscellaneous issues.

- **False Detection**: Detecting a wrong object as the target category.
- **Missed Detection**: Failure to detect the object even though it was in view.
- **Loop**: Poor exploration due to looping over the same locations.
- **Trapped**: Repeated collisions with the same or nearby objects cause the agent getting trapped in coverage. Includes when the agent is trapped in spawn.
- **Explore**: Generic failures to find the target object although the agent explores new areas efficiently. Includes semantic failures e.g. going outdoors to find a bed.
- **Attention**: Despite detecting the target object, the agent ignores and does not navigate to the seen object.
- **Last mile**: Failure to navigate to the detected target object.

- **Commitment**: Continues past the detected target object.
- **Stairs**: The target is on a different floor and the agent is unable to navigate up/down stairs.
- **Misc**: Other causes of failure such as scene mesh artifacts, the agent randomly quitting the episode, etc.

These failures modes are based off those presented in EEAux [24] when analyzing their agent. However, Ye et al. (2021) [24] only do so for agents with ground-truth semantic information. In practice, the agent will not have access to this information, so previous analysis does not provide the full picture on how the agent would perform in object navigation. We not only replicate the analysis with ground-truth information, but also investigate the failure modes of the more practical agent that predicts semantics using a pretrained semantic segmentation model.

In addition, we compute a 95% confidence interval using bootstrapping [8].

## 3.1 Analysis of EEAux

We collected and analyzed 300 validation episodes using the pretrained weights released by Ye et al [24]. The breakdown of failure modes can be found in Table 3.1.

We find that when using semantic predictions of the scene, false positives of the target object comprise the largest individual failure mode, accounting for over a third of all failures. Indeed false positives can have a significant impact on the agent's behavior, leading it to plan specific paths to move towards the (incorrectly) detected object and, most impactful, stop and end the episode thinking that it has found the target. In addition, we observed a smaller proportion of failures in which the semantic segmentation model failed to detect the target object despite seeing it, which led the agent to miss the target and continue exploring the environment. The second most prevalent type of failures are those related to exploration, accounting for about a third of failures in total. The remaining failure modes each account for a small percentage of the failures, with the *misc* mode accounting a remaining 21.9%.

| Failure Mode | EEAux % | EEAux GT % |
|---|---|---|
| **False Detection** | $\mathbf{35.3 \pm 6.3}$ | $1.4 \pm 1.9$ |
| **Missed Detection** | $6.3 \pm 3.2$ | $1.4 \pm 1.9$ |
| **Loop** | $14.7 \pm 4.6$ | $22.2 \pm 6.8$ |
| **Trapped** | $12.5 \pm 4.3$ | $13.9 \pm 5.6$ |
| **Explore** | $8.9 \pm 3.7$ | $12.5 \pm 5.4$ |
| **Attention** | $2.2 \pm 1.9$ | $3.5 \pm 3.0$ |
| **Last Mile** | $2.7 \pm 2.1$ | $3.5 \pm 3.0$ |
| **Commitment** | $0.9 \pm 1.2$ | $2.8 \pm 2.7$ |
| **Stairs** | $0.4 \pm 0.9$ | $6.9 \pm 4.2$ |
| **Misc** | $21.9 \pm 5.4$ | $31.9 \pm 7.6$ |
| **Success Rate** | 25.3 | 51.7 |

Table 3.1: Occurrence of failure modes of EEAux [24] over 300 validation episodes. Each row reports the percentage of failures attributed to a particular cause along with 95% confidence intervals. The primary cause of failure is false positives in object detection followed by exploration related issues of *loop*, *trapped* and *explore*. The second column shows how these numbers change when the agent is provided with ground truth, instead of predicted, semantic maps. The last row reports the overall success rate.

We also analyzed EEAux when it was provided ground-truth semantic information. Unsurprisingly, the agent suffered less from failures in object detection, accounting for only about 3% of all failures. We did observe a couple failures due to an error in the ground-truth semantics, as shown in Figure 3-1. The remainder of failures remain distributed across the failure modes in a similar way to the proportions with predicted semantics, with exploration-related failures comprising a majority of the remaining failures and a small rate of other non-misc failure modes. Note that while the percentages have increased as object detection is not an issue with the ground-truth, the number of failures has remained roughly the same, with the increase in success rate being almost entirely accounted for by the mitigation of the object detection failure modes. This indicates that EEAux suffers from issues related to exploration and navigating to found targets that are disjoint for any issues with object detection.

As we found object detection to cause the most failures, we focused our work on first improving this component of the agent. To this end, we propose an additional module to the end of the policy that verifies if the given observations are indicative

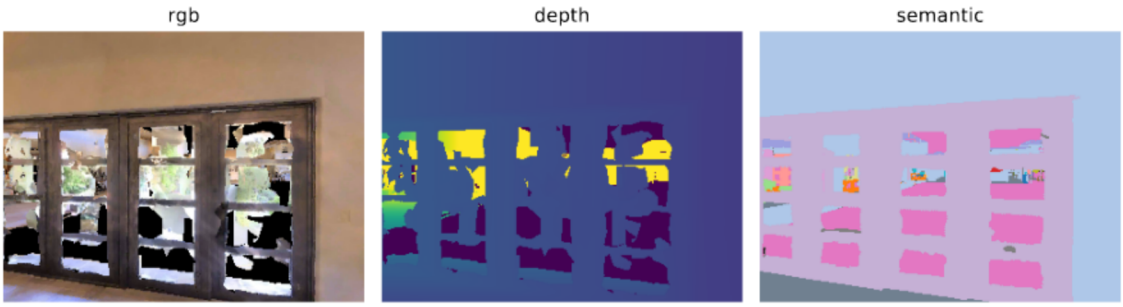



Figure 3-1: RGB, depth, and semantics for a view in a MP3D validation scene in which semantics are incorrect. The ground outside the window is erroneously labelled as a bed (dark pink), which leads to the agent moving to the window and stopping when looking for a bed.

of success and overrides the *stop* action outputted by the policy net if a false success is detected, similar to the verifier introduced in a map-based approach in [16]. Key to this module is a success classifier, which determines with the *stop* action should be accepted or overridden. We describe our approach to implementing such a classifier in the next section.

# Chapter 4

# Method

## 4.1 Data

As we plan to apply our work to the object navigation task, we use the Matterport3D dataset (MP3D [4]) to generate image data for fine-tuning our success classifier models. For success verification, we generate positive examples by sampling viewpoints for labelled target objects in MP3D. To generate negative examples, we run EEAux [24] with predicted semantics to collect training episodes and use the last frame from each unsuccessful episode. This not only provides which generic examples where the target object is missing, but also examples in which the agent failed due to falsely detecting the target object, both of which we want our success classifiers to be accurate on. In both cases, we only use scenes from the training split of MP3D, to be able to test the generalizability of introducing a success verification module to the agent. Overall, we generate a dataset of about 5500 examples, comprised of about 4000 positive and 1500 negative examples. The breakdown of examples by target object can be found in Table 4.1. Note that more common objects occur more in the dataset, as both the viewpoint sampling and validation episodes scale with the number of target instances.

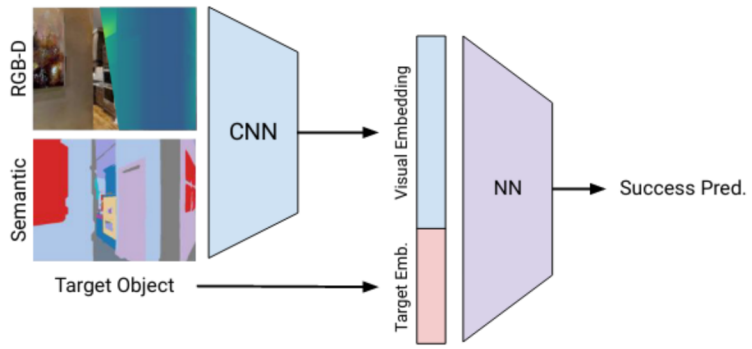
For training semantic segmentation models, we follow previous work [24, 16] in collecting about 100K random views evenly from the training MP3D scenes (1792 per scene). We split the scenes into a 80/20 train-val split. We end up with 80640 training examples and 19712 validation examples.

| Target Object | Train Examples | Val Examples |
|---|---|---|
| **bathtub** | 56 | 20 |
| **bed** | 141 | 40 |
| **cabinet** | 333 | 72 |
| **chair** | 919 | 223 |
| **chest of drawers** | 184 | 40 |
| **clothes** | 35 | 9 |
| **counter** | 117 | 35 |
| **cushion** | 534 | 124 |
| **fireplace** | 56 | 12 |
| **gym equipment** | 9 | 5 |
| **picture** | 252 | 68 |
| **plant** | 223 | 55 |
| **seating** | 101 | 35 |
| **shower** | 121 | 28 |
| **sink** | 193 | 64 |
| **sofa** | 130 | 35 |
| **stool** | 132 | 50 |
| **table** | 548 | 123 |
| **toilet** | 140 | 33 |
| **towel** | 163 | 32 |
| **tv monitor** | 46 | 6 |
| **Overall** | 4433 | 1109 |

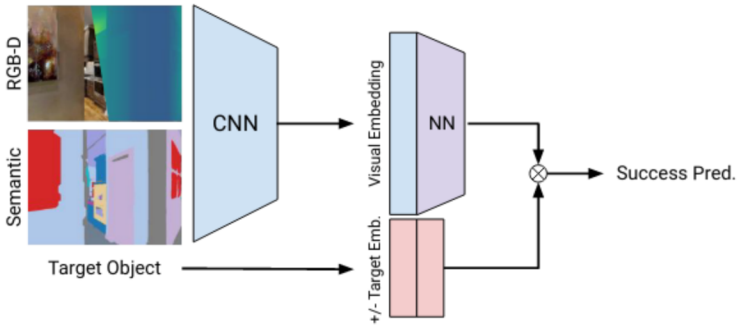
Table 4.1: Per-object breakdown of training and validation examples in our generated success verification dataset, along with total counts.

## 4.2 Success Verification

We define the success verification problem as follows: given a set of egocentric image observations and a target object category, predict whether our location is a success or not. The model will have access to RGB and depth observations, as well as ground-truth semantic masks in some experiments. As our dataset is not balanced, we train using a class-weighted cross-entropy loss (see Appendix A for exact weights) and evaluate using balanced accuracy, i.e. the average of the accuracies for each class. For semantic segmentation, we use a RedNet model [11] with the pretrained weights provided by [24] by finetuning on 100K randomly sampled views from MP3D.




(a) Model with embedding concatenation




(b) Model with embedding dot product

Figure 4-1: Model architectures for each form of embedding aggregation. In (a), the output of the CNN visual encoder is concatenated with a learned embedding of the target object and passed through a final neural network. In (b), the dot product is taken between the output of the visual encoder and learned positive and negative embeddings of the target object to compute the output logits.

### 4.2.1 Learning-based Classifier

We consider two main architectures for the success classifier, shown in Figure 4-1, distinguished by their methods for aggregating the visual observation and target object embeddings for downstream prediction. In the concat model (Figure 4-1a), we concatenate these two embeddings and pass it through a neural network that outputs the logits for each class. For our work, we use a single linear layer. The second dot product architecture, we instead have two embeddings per object category, corresponding to the positive and negative classes, which we then take dot products with

a compressed visual embedding to get the corresponding logits. We implement the compression network as a single linear layer. Utilizing a dot product for embedding comparison aligns well with work in other fields, such as in few shot learning where approaches like prototypical networks [20] learn a metric space for classification by comparing test examples to class prototype representations. For both architectures, we use a ResNet model [10] with the weights from the visual encoder of the policy network from the baseline [24]. For the semantic input, we pass in a semantic mask that labels each pixel with a single class. Each object label is mapped to an embedding, with the embedding dimension acting as the channel dimension. This is then concatenated with the RGB-D image along the channel dimension and then passed through the ResNet model. We present the results of our comparison of the two architectures in Section 5.2.

In addition, we investigate three different ways to convert predicted semantic logits into an input for the success classifier. The first two convert the logits into a semantic mask with a class per pixel, either through an argmax of the logits or by sampling the probability distribution given by a softmax activation. In the latter, we average multiple success predictions by sampling multiple masks from the predicted logits. The semantic mask is then passed through a pretrained embedding layer, mapping each class to an embedding vector. For the third method, we use the probability distribution as coefficients and compute a linear combination of semantic embedding weights per pixel. This method takes inspiration from previous work like mixup [25] that also mix together different inputs, although we do so in hopes of interpolating results from uncertain predictions rather than to improve generalization by encouraging interpolation. The results for these experiments are in Section 5.3.

### 4.2.2 Heuristic Baseline

Along with learning-based models for predicting success, we also consider a logistic regression classifier. Specifically, this classifier takes semantic images and computes the percentage of the image that contains the target object. Ye et al. (2021) used this feature, known as “Semantic Goal Exists" (SGE), and ablation studies show that

the feature had a significant impact on the success and SPL of the agent [24]. We hypothesize that this feature should also work well to determine success, as it tells both whether the target object is in view and roughly indicates proximity to target object—a larger SGE correlates with being closer to the target object, as closer objects will take more of the agent's view. Effectively, we are using SGE as a heuristic for success, finding a threshold determined by a learned weight and offset from logistic regression on the training data.

When using predicted semantics, we consider two methods to compute SGE: 1) taking the argmax of logits per pixel and computing the percentage of pixels labelled as the target object class, and 2) computing the mean of the predicted probability for the target object. The second approach retains the relative uncertainties of predictions, as more certainty leads to higher probabilities and therefore a higher mean. We hypothesize that this additional granularity maybe useful, as more certainty should strengthen the confidence of the success prediction. We train a L2-regularized logistic regression model per target object category and compute the balanced accuracy for each and in total. We use a balanced weighting of the classes in the loss function to account for class imbalance. We present these results in Section 5.1.

## 4.3 Ensembling Model

In addition to a RedNet model [11] for semantic segmentation, we also train an ensemble model by independently training 5 heads from the RedNet last layer output, implemented as 2D transposed convolutional layers. We use the pretrained weights from [24] for the frozen RedNet, aligning with our single non-ensemble predictions. We train this model for 100 epochs. Further implementation details can be found in Appendix A and results for the ensemble model for success verification are presented in Section 5.4.

# Chapter 5

# Results

## 5.1 SGE Baseline

We train a logistic regression for each target object category and compute their individual and collective accuracies, accuracies on each class, and balanced accuracies (see Table 5.1). When using ground-truth semantics, we find that the baseline achieves a balanced accuracy of 50.2 on the validation set, performing slightly better than a random classifier which would achieve a balanced accuracy of 50. In addition, we trained a single logistic regression fitted to the entire training dataset and evaluated on the entire validation dataset, and found that our per-object model outperformed the model by 4.6%. We conjecture that this may partly be due to the per-object model's ability to align its weights with the specific target object—larger objects will have a larger SGE when in view than smaller objects from the same distance, so the threshold for success may differ among the categories. A full breakdown of accuracies by target object can be found in Table 5.1. Note that the poor performance on the gym equipment class is partially attributed to the lack of data, including a lack of negative examples in the validation set, so we mostly ignore the target object in our discussion here.

When using predicted semantics, we consider two methods for computing an SGE feature, as described in Section 4.2.2, and report their results in Table 5.2. Overall, we found that using prediction probabilities performed slightly better between to

| Target Object | Accuracy | Pos. Acc. | Neg. Acc. | Balanced Accuracy |
|---|---|---|---|---|
| **bathtub** | 40.0 | 0.0 | 100.0 | 50.0 |
| **bed** | 65.0 | 75.0 | 41.7 | 58.3 |
| **cabinet** | 66.7 | 94.0 | 4.5 | 49.3 |
| **chair** | 59.2 | 69.5 | 22.4 | 46.0 |
| **chest of drawers** | 70.0 | 93.1 | 9.1 | 51.1 |
| **clothes** | 33.3 | 14.3 | 100.0 | 57.1 |
| **counter** | 71.4 | 100.0 | 0.0 | 50.0 |
| **cushion** | 62.9 | 82.6 | 6.2 | 44.4 |
| **fireplace** | 25.0 | 0.0 | 50.0 | 25.0 |
| **gym equipment** | 0.0 | 0.0 | 0.0 | 0.0 |
| **picture** | 60.3 | 86.7 | 8.7 | 47.7 |
| **plant** | 27.3 | 2.4 | 100.0 | 51.2 |
| **seating** | 48.6 | 100.0 | 0.0 | 50.0 |
| **shower** | 78.6 | 100.0 | 14.3 | 57.1 |
| **sink** | 79.7 | 98.0 | 20.0 | 59.0 |
| **sofa** | 17.1 | 10.0 | 60.0 | 35.0 |
| **stool** | 82.0 | 93.2 | 0.0 | 46.6 |
| **table** | 47.2 | 25.0 | 88.4 | 56.7 |
| **toilet** | 51.5 | 100.0 | 5.9 | 52.9 |
| **towel** | 50.0 | 31.8 | 90.0 | 60.9 |
| **tv monitor** | 50.0 | 100.0 | 0.0 | 50.0 |
| **Overall** | 57.5 | 67.1 | 33.2 | 50.2 |
| **Single Reg.** | 62.8 | 80.0 | 16.0 | 48.0 |

Table 5.1: Validation accuracy and balanced accuracy of SGE baseline using ground-truth semantics on each target object as well as overall. We also compare these results to a single logistic regression trained over all of the data, labelled "Single Reg."

two methods. We conjecture that this may partially be due to the finer granularity in its computed SGE feature, which mitigates the impact of incorrect predictions. In contrast, argmax uses more of an all-or-nothing approach, so a false positive or false negative can have a much more significant impact on the overall SGE. The full results for argmax can be found in Table B.1. We also note that among the individual logistic regression models, more of them are accurate on only one class in the argmax formulation. Notably, 12 of the 21 are perfectly accurate on one class and perfectly inaccurate on the other, as opposed to only 4 when using ground-truth semantics and zero when using predicted probabilities. Also, we again found that the per-object model outperformed a single logistic regression over the entire dataset.

Finally, we also experiment on the two methods of computing SGE to train a per-

| Target Object | Accuracy | Pos. Acc. | Neg. Acc. | Balanced Accuracy |
|---|---|---|---|---|
| **bathtub** | 75.0 | 83.3 | 62.5 | 72.9 |
| **bed** | 42.5 | 32.1 | 66.7 | 49.4 |
| **cabinet** | 59.7 | 64.0 | 50.0 | 57.0 |
| **chair** | 55.6 | 58.6 | 44.9 | 51.8 |
| **chest of drawers** | 55.0 | 58.6 | 45.5 | 52.0 |
| **clothes** | 66.7 | 85.7 | 0.0 | 42.9 |
| **counter** | 45.7 | 44.0 | 50.0 | 47.0 |
| **cushion** | 37.1 | 25.0 | 71.9 | 48.4 |
| **fireplace** | 58.3 | 83.3 | 33.3 | 58.3 |
| **gym equipment** | 0.0 | 0.0 | 0.0 | 0.0 |
| **picture** | 54.4 | 66.7 | 30.4 | 48.6 |
| **plant** | 61.8 | 70.7 | 35.7 | 53.2 |
| **seating** | 60.0 | 35.3 | 83.3 | 59.3 |
| **shower** | 67.9 | 90.5 | 0.0 | 45.2 |
| **sink** | 65.6 | 65.3 | 66.7 | 66.0 |
| **sofa** | 60.0 | 56.7 | 80.0 | 68.3 |
| **stool** | 48.0 | 40.9 | 100.0 | 70.5 |
| **table** | 39.8 | 22.5 | 72.1 | 47.3 |
| **toilet** | 57.6 | 56.2 | 58.8 | 57.5 |
| **towel** | 75.0 | 86.4 | 50.0 | 68.2 |
| **tv monitor** | 50.0 | 0.0 | 100.0 | 50.0 |
| **Overall** | 53.1 | 51.8 | 56.5 | 54.2 |
| **Argmax** | 51.1 | 48.4 | 58.1 | 53.3 |
| **Single Reg.** | 41.3 | 70.1 | 26.8 | 48.5 |

Table 5.2: Validation accuracy, accuracy on each class, and balanced accuracy of SGE baseline using RedNet predicted semantic probabilities on each target object as well as overall. We also compare these results to a model trained on the argmax variant of SGE and a single logistic regression trained over all of the data, labelled "Single Reg."

object logistic regression using the ensemble semantic segmentation model. Similar to our results with the single RedNet model, we found that computing SGE with the predicted probabilities (see Table 5.3 tended to perform better, with a similar 6.7% increase in balanced accuracy. The full results for the argmax variant can be found in Table B.2. Unlike our other SGE baseline results, we found that a single logistic regression fitted to the entire dataset performed better in this case.

| Target Object | Accuracy | Pos. Acc. | Neg. Acc. | Balanced Accuracy |
|---|---|---|---|---|
| **bathtub** | 75.0 | 83.3 | 62.5 | 72.9 |
| **bed** | 42.5 | 32.1 | 66.7 | 49.4 |
| **cabinet** | 59.7 | 64.0 | 50.0 | 57.0 |
| **chair** | 55.6 | 58.6 | 44.9 | 51.8 |
| **chest of drawers** | 55.0 | 58.6 | 45.5 | 52.0 |
| **clothes** | 66.7 | 85.7 | 0.0 | 42.9 |
| **counter** | 45.7 | 44.0 | 50.0 | 47.0 |
| **cushion** | 37.1 | 25.0 | 71.9 | 48.4 |
| **fireplace** | 58.3 | 83.3 | 33.3 | 58.3 |
| **gym equipment** | 0.0 | 0.0 | 0.0 | 0.0 |
| **picture** | 54.4 | 66.7 | 30.4 | 48.6 |
| **plant** | 61.8 | 70.7 | 35.7 | 53.2 |
| **seating** | 60.0 | 35.3 | 83.3 | 59.3 |
| **shower** | 67.9 | 90.5 | 0.0 | 45.2 |
| **sink** | 65.6 | 65.3 | 66.7 | 66.0 |
| **sofa** | 60.0 | 56.7 | 80.0 | 68.3 |
| **stool** | 48.0 | 40.9 | 100.0 | 70.5 |
| **table** | 39.8 | 22.5 | 72.1 | 47.3 |
| **toilet** | 57.6 | 56.2 | 58.8 | 57.5 |
| **towel** | 75.0 | 86.4 | 50.0 | 68.2 |
| **tv monitor** | 50.0 | 0.0 | 100.0 | 50.0 |
| **Overall** | 51.9 | 51.6 | 52.7 | 52.2 |
| **Argmax** | 40.8 | 30.2 | 67.7 | 48.9 |
| **Single Reg.** | 45.3 | 52.4 | 55.6 | 54.0 |

Table 5.3: Validation accuracy, accuracy on each class, and balanced accuracy of SGE baseline using ensemble predicted semantic probabilities on each target object as well as overall. We also compare these results to a model trained on the argmax variant of SGE and a single logistic regression trained over all of the data, labelled "Single Reg."

## 5.2 Image-Goal Feature Aggregation Methods

As discussed in our Section 4.2.1, we compared two methods to aggregate the visual features and target embedding within the success classifier; namely, to concatenate the two and pass them through a neural network (a linear layer in our case) or to compute the dot product between the visual features and target embedding. We trained models using each method for 300 epochs using the ground-truth semantic segmentation labels and plotted their balanced accuracy in Figure 5-1. Note that while dot product aggregation significantly outperformed concatenation in training,

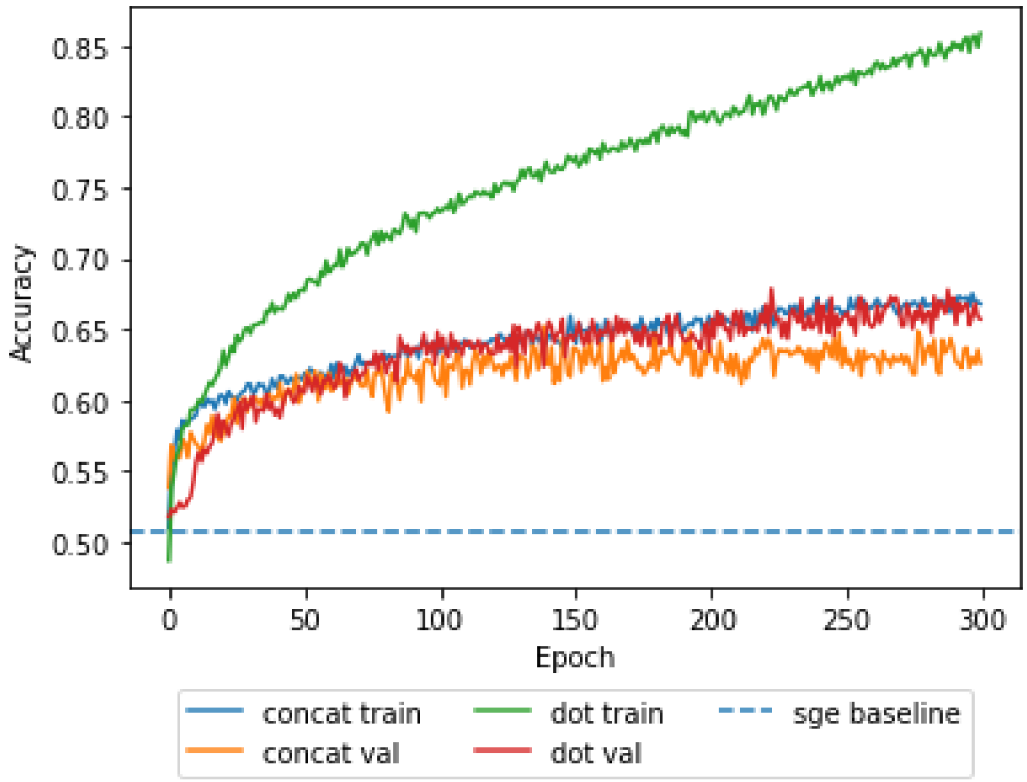



Figure 5-1: Balanced accuracies of the concatenation and dot product models on the train and validation sets.

some of this increase could be explained by overparameterization, as the dot product model effectively learns a separate set of weights (i.e. the target embedding) per target category, whereas the concatenation model must learn a linear head that generalizes to all target categories. However, we still find that dot product aggregation outperforms concatenation on the validation set, and use this mode for the all models in sections below.

## 5.3 Predicted Semantics

In practice, agents will not have access to ground-truth semantic information in object navigation, as agents are placed in unseen environments at test time. In this section, we continue our experiments by using predicted semantics. As shown in Figure 5-2, we suffer a significant drop in balanced accuracy, due to the decrease in

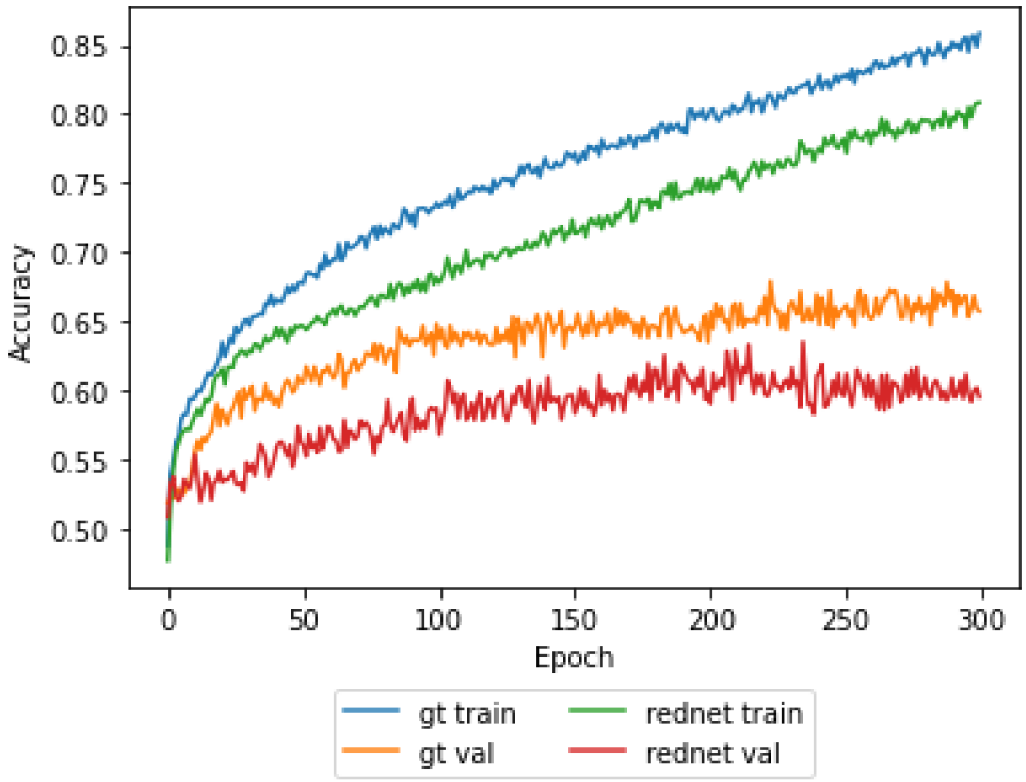



Figure 5-2: Balanced accuracies of classifiers trained using ground-truth and predicted semantics. We use the pretrained RedNet weights from [24] for prediction.

accuracy in the semantic input. In addition, we try initializing our classifier weights to that of the trained ground-truth model from Section 5.2, but find that while we see improved validation accuracy at the onset of training, the two models plateau to similar accuracies at convergence as shown in Figure 5-3.

Figure 5-4 compares the balanced training and validation accuracies of our success classifier model using three methods of converting semantic predictions into inputs. We trained the models that use argmax and a weighted sum of pretrained embeddings for 300 epochs, whereas we trained the sampling based method for 200 epochs due to the increased overhead due to having multiple forward passes per prediction. We find that all methods are comparable to each other, with accuracies tracking each other through all epochs.

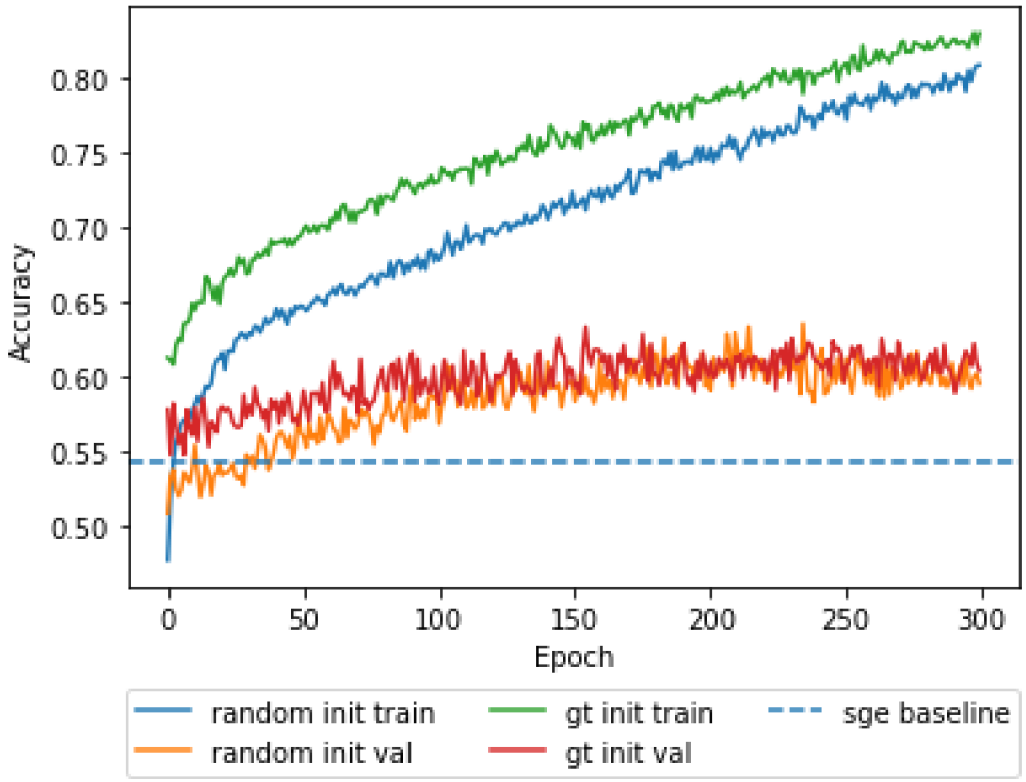



Figure 5-3: Balanced accuracies of classifiers trained using predicted semantics initialized randomly or using the trained ground-truth model.

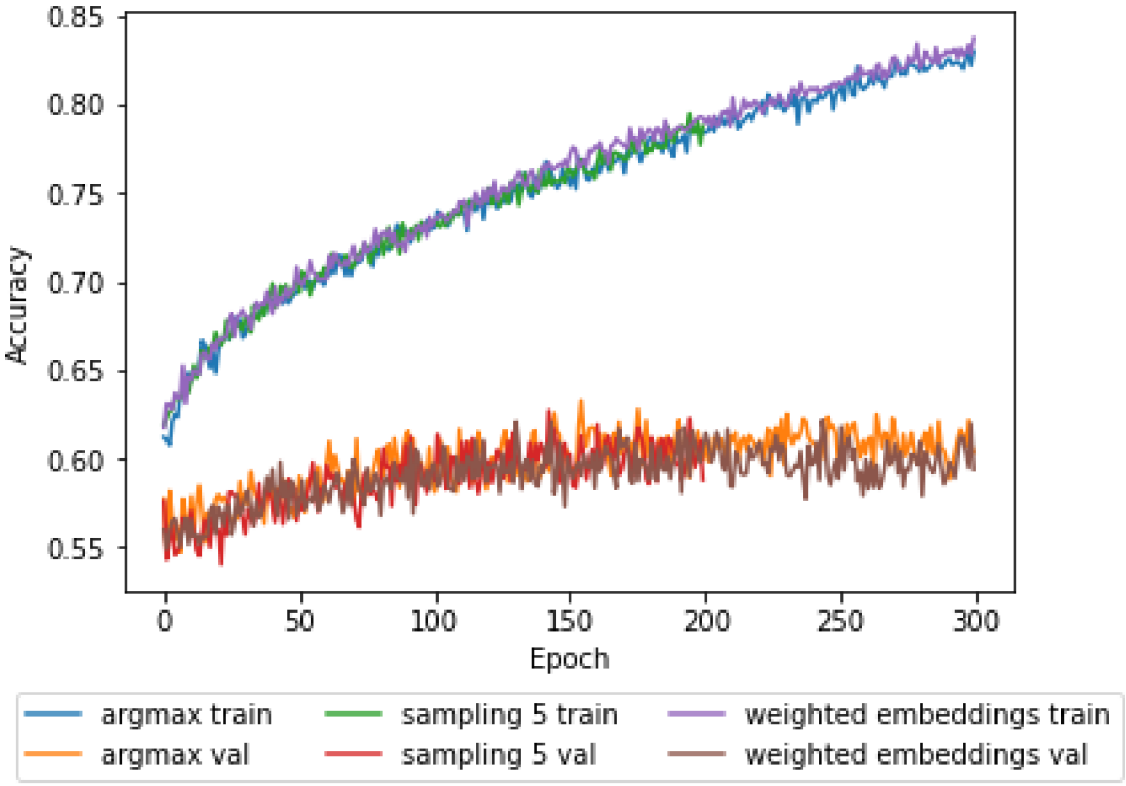



Figure 5-4: Balanced accuracies of classifiers using different methods to produce semantic inputs. In *sampling5*, five inputs are sampled from the predicted probabilities and the success prediction logits are averaged. In *weighted embeddings*, a linear combination of the predicted probabilities and embeddings from the embedding layer in our pretrained visual encoder are computed. We trained all models for 300 epochs except the sampling based method, which was trained for 200 due to the increased time overhead from multiple forward passes.

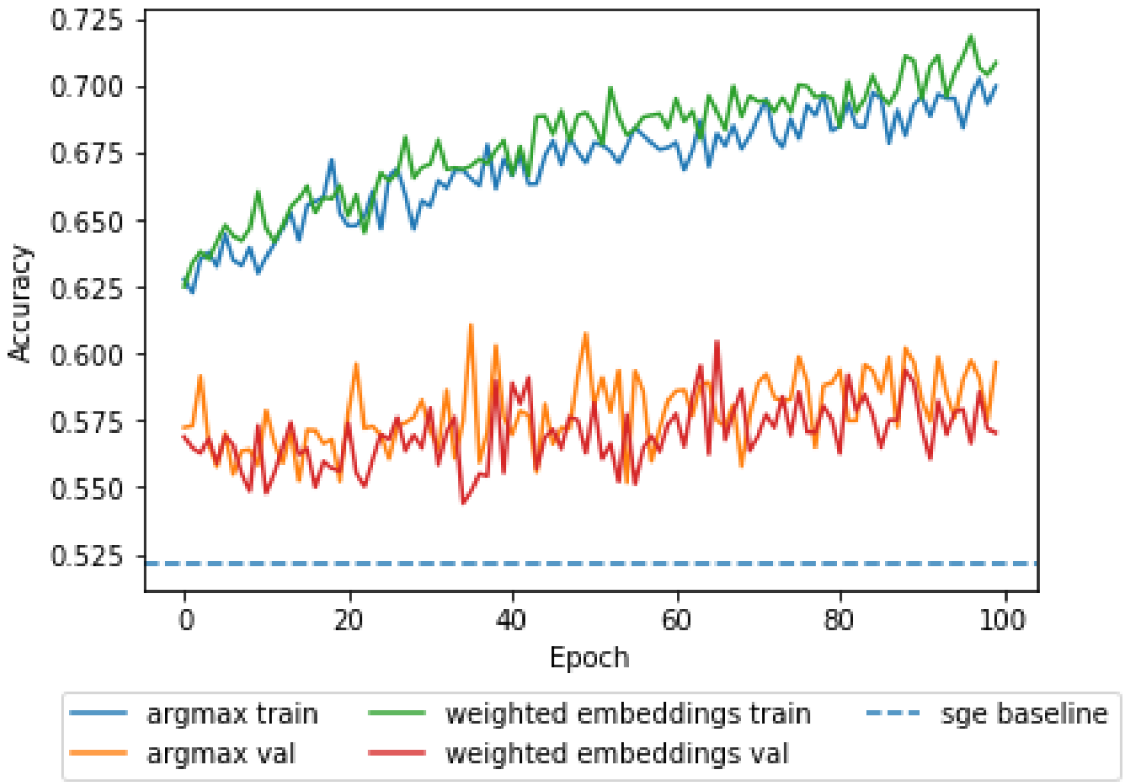



Figure 5-5: Balanced accuracies of classifiers trained using ensemble predicted semantics. We use the pretrained RedNet weights from [24] for prediction and train 5 heads on top using 100K randomly sampled views from MP3D.

## 5.4 Ensemble Model

Finally, we replace the original RedNet model with an ensemble model for semantic segmentation. We use the average of the probability outputs for prediction, and investigate the impact of different methods to produce semantic inputs on the performance of the trained downstream success classifier. We omit the sampling method as we found it to be more time and memory intensive. Like with the single RedNet model, we found that these methods perform comparable to each other, as shown in Figure 5-5. In addition, as shown in Figure 5-6, we found that the success classifier trained using semantic predictions from the ensemble model performs about the same if not slightly worse than the same method using a single RedNet.

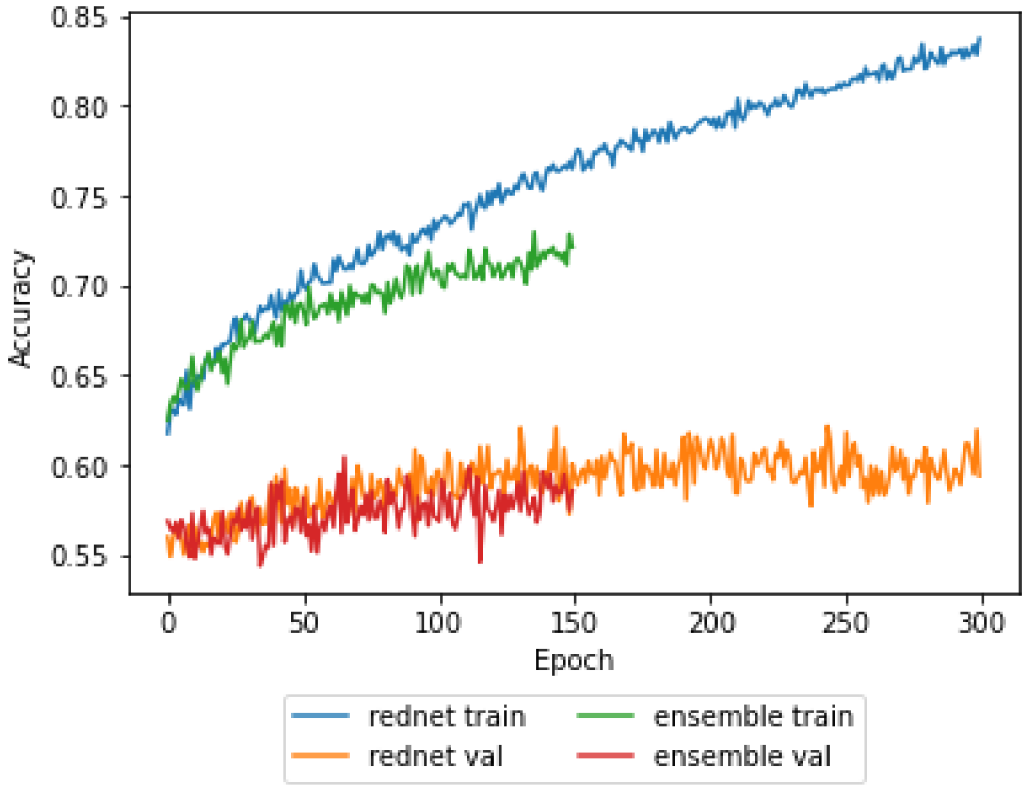



Figure 5-6: Balanced accuracies of classifiers trained using single RedNet and ensemble predicted semantics using the weighted embedding method to produce semantic inputs.

# Chapter 6

# Conclusion

In this work, we identify false positives in object detection to be the main mode of failure for current state-of-the-art agents for object navigation. We propose a method to combat false positives by verifying assumed success by the policy network (i.e. when the policy outputs the *stop* action) and overriding the action to continue the episode if verification fails. In this work, we primarily work on the main component for such a module: a classifier to determine success from the agent's egocentric visual observations. We find that learning a joint visual-semantic embedding space and comparing the visual and target object representations performs well at classifying success in object navigation. We also did some preliminary investigation into using ensembling to improve the semantic segmentation model, but find our current approach to underperform the original RedNet model.

## 6.1 Future Work

### 6.1.1 Improving Success Verification

As discussed, we plan to implement a success verification module in our agent using the best performing success classifier. The module would activate if the policy network outputs a *stop* action and if the classifier identifies the current observation as a false positive, the agent's actions for the next few timesteps will be replaced by a series of

turn actions to orient the agent away from the offending viewpoint.

We also expect that we could iterate and improve the dataset we generate for success verification. Note that while in our success verification dataset all positive examples have the target object in view, in the object navigation task as defined by Habitat [3] we do not necessarily have to end with the target in view, only that the agent is in a position such that there is a camera orientation in which the target is in view. However, we expect expanding positive examples to such situations would lead to many similar images in both classes, making the decision boundary harder to find. As we mainly wanted to use this success classifier to combat false positives, we expect a stricter set of positive examples to help with this. In practice, we may find that a result in between combating false positives and false negatives to work better, which would be the interest of future work.

Finally, we would like to extend our work with ensemble models, as we found our results with the ensemble model to fall below that of a single RedNet model. We would like to continue tuning our training of the ensemble last layers, as we generally found RedNet to require a large amount of GPU RAM for even a single $480 \times 640$ image. In addition, we could likely improve the ensemble model by using more recent techniques, such as adding an adversarial loss as proposed in [14].

### 6.1.2 Further Improvements to Semantic Understanding

In addition, we expect that improvements to semantic segmentation inputs to the policy network would improve performance on object navigation. Related, Chaplot et al (2021) [5] found that by maximizing segmentation confidence in exploration via an intrinsic reward and then finetuning using images of MP3D scenes using labels generated from the most confident predictions improves upon previous state-of-the-art in map-based object navigation agents. This method helps address three concerns: 1) overcoming the distribution shift between Internet-curated images and embodied observations, as the latter may not fully frame visible objects; 2) avoid uncertain predictions by finetuning to high confidence labels; and 3) enable the first two even if ground-truth labels are not available. Therefore, we hypothesize that techniques

to improve prediction accuracy and quantify uncertainty would also improve end-to-end learning-based solutions. In particular, we expect that we could apply the same ensemble training and proposed extensions [14] to improve semantic segmentation predictions that are passed to the policy network. We may also be able to draw upon more recent work in uncertainty prediction, such as deep evidential networks [1].

# Appendix A

# Implementation Details

For the SGE baselines, we train a L2-regularized logistic regression per target object, with a regularization coefficient of 0.1 and each class-specific loss term is weighed by the inverse of the proportion of the dataset that is class $i$.

We train our success classifiers using gradient descent with cross-entropy loss, the Adam optimizer [13], and a batch size of 128 on a single Volta V100. Due to the class imbalance in our generated dataset, we reweigh each term by $1 - P_i$ where $P_i$ is the proportion of the dataset that is class $i$. We use a learning rate of $3 \times 10^{-4}$ for all training procedures. When using ensemble semantic segmentation models, in which we use micro-batching of size 8 due to CUDA memory limitations, we also tried scaling the learning rate down by a factor of $\frac{128}{8} = 16$ accordingly, but found training results to be comparable if not worse, as shown in A-1. Also we use an embedding dimension of 4 when embedding semantic masks, aligning with the visual encoder used by [24].

We train the ensemble semantic segmentation model using cross-entropy loss, the Adam optimizer, and a batch size of 4 on 2 Volta V100 GPUs. We found it generally difficult to increase the batch size per GPU, as we found RedNet to require a large amount of CUDA RAM for training (on the order of over 5 GB per batch). We use an learning rate of $2 \times 10^{-3}$, a momentum of 0.9, and weight decay of $10^{-4}$.

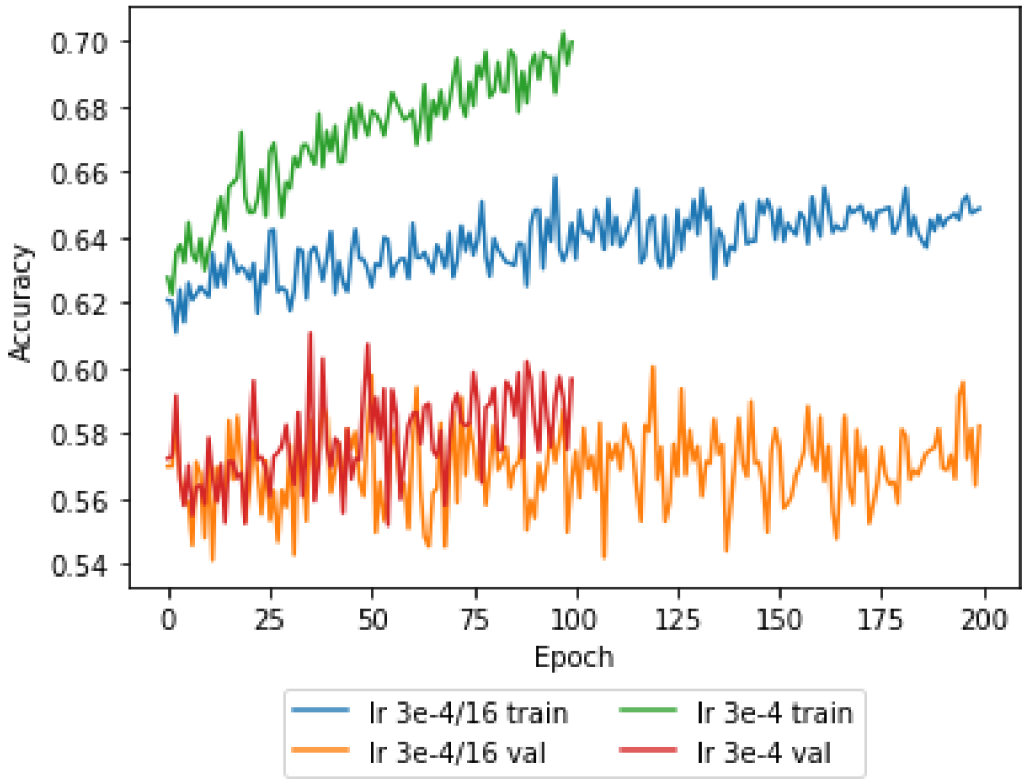



Figure A-1: Balanced accuracies of classifiers using ensemble semantic predictions for different learning rates

# Appendix B

# SGE Baseline results for the argmax variant of SGE

Included here are the full results on the SGE baselines where SGE was computed by finding the proportion of pixels labelled as the target object after taking the argmax of predicted semantics per pixel.

| Target Object | Accuracy | Pos. Acc. | Neg. Acc. | Balanced Accuracy |
|---|---|---|---|---|
| **bathtub** | 40.0 | 0.0 | 100.0 | 50.0 |
| **bed** | 30.0 | 0.0 | 100.0 | 50.0 |
| **cabinet** | 30.6 | 0.0 | 100.0 | 50.0 |
| **chair** | 71.3 | 86.8 | 16.3 | 51.6 |
| **chest of drawers** | 57.5 | 55.2 | 63.6 | 59.4 |
| **clothes** | 66.7 | 85.7 | 0.0 | 42.9 |
| **counter** | 28.6 | 0.0 | 100.0 | 50.0 |
| **cushion** | 73.4 | 98.9 | 0.0 | 49.5 |
| **fireplace** | 50.0 | 0.0 | 100.0 | 50.0 |
| **gym equipment** | 0.0 | 0.0 | 0.0 | 0.0 |
| **picture** | 58.8 | 88.9 | 0.0 | 44.4 |
| **plant** | 25.5 | 0.0 | 100.0 | 50.0 |
| **seating** | 57.1 | 29.4 | 83.3 | 56.4 |
| **shower** | 75.0 | 100.0 | 0.0 | 50.0 |
| **sink** | 78.1 | 95.9 | 20.0 | 58.0 |
| **sofa** | 14.3 | 0.0 | 100.0 | 50.0 |
| **stool** | 12.0 | 0.0 | 100.0 | 50.0 |
| **table** | 35.8 | 10.0 | 83.7 | 46.9 |
| **toilet** | 51.5 | 0.0 | 100.0 | 50.0 |
| **towel** | 31.2 | 0.0 | 100.0 | 50.0 |
| **tv monitor** | 50.0 | 0.0 | 100.0 | 50.0 |
| **Overall** | 51.1 | 48.4 | 58.1 | 53.3 |
| **Single Reg.** | 32.2 | 78.5 | 20.1 | 49.3 |

Table B.1: Validation accuracy, accuracy on each class, and balanced accuracy of SGE baseline using the argmax of RedNet predicted semantics on each target object as well as overall. We also compare these results to a single logistic regression trained over all of the data, labelled "Single Reg."

| Target Object | Accuracy | Pos. Acc. | Neg. Acc. | Balanced Accuracy |
|---|---|---|---|---|
| **bathtub** | 75.0 | 83.3 | 62.5 | 72.9 |
| **bed** | 42.5 | 32.1 | 66.7 | 49.4 |
| **cabinet** | 59.7 | 64.0 | 50.0 | 57.0 |
| **chair** | 55.6 | 58.6 | 44.9 | 51.8 |
| **chest of drawers** | 55.0 | 58.6 | 45.5 | 52.0 |
| **clothes** | 66.7 | 85.7 | 0.0 | 42.9 |
| **counter** | 45.7 | 44.0 | 50.0 | 47.0 |
| **cushion** | 37.1 | 25.0 | 71.9 | 48.4 |
| **fireplace** | 58.3 | 83.3 | 33.3 | 58.3 |
| **gym equipment** | 0.0 | 0.0 | 0.0 | 0.0 |
| **plant** | 61.8 | 70.7 | 35.7 | 53.2 |
| **seating** | 60.0 | 35.3 | 83.3 | 59.3 |
| **shower** | 67.9 | 90.5 | 0.0 | 45.2 |
| **sink** | 65.6 | 65.3 | 66.7 | 66.0 |
| **sofa** | 60.0 | 56.7 | 80.0 | 68.3 |
| **stool** | 48.0 | 40.9 | 100.0 | 70.5 |
| **table** | 39.8 | 22.5 | 72.1 | 47.3 |
| **toilet** | 57.6 | 56.2 | 58.8 | 57.5 |
| **towel** | 75.0 | 86.4 | 50.0 | 68.2 |
| **tv monitor** | 50.0 | 0.0 | 100.0 | 50.0 |
| **Overall** | 40.8 | 30.2 | 67.7 | 48.9 |
| **Single Reg.** | 48.2 | 91.0 | 10.2 | 50.6 |

Table B.2: Validation accuracy, accuracy on each class, and balanced accuracy of SGE baseline using the argmax of ensemble predicted semantics on each target object as well as overall. We also compare these results to a single logistic regression trained over all of the data, labelled "Single Reg."

# Bibliography


[1] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In *NeurIPS*, 2020.

[2] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[3] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[5] Devendra Singh Chaplot, Murtaza Dalal, Saurabh Gupta, Jitendra Malik, and Ruslan Salakhutdinov. Seal: Self-supervised embodied active learning. In *NeurIPS*, 2021.

[6] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *NeurIPS*, 2020.

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Kumar Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *ICLR*, 2020.

[8] Bradley Efron and Robert Tibshirani. The bootstrap method for assessing statistical accuracy. *Behaviormetrika*, 12(17):1–35, 1985.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:1806.01054*, 2018.

[12] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *arXiv preprint arXiv:2111.09888*, 2021.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[14] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

[15] Yifan Liu, Chunhua Shen, Changqian Yu, and Jingdong Wang. Efficient semantic video segmentation with per-frame inference. In *ECCV*, 2020.

[16] Haokuan Luo, Albert Yue, Zhang-Wei Hong, and Pulkit Agarwal. Stubborn: A strong baseline for indoor object navigation. *arXiv preprint arXiv:2203.07359*, 2022.

[17] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. In *ICCV*, 2021.

[18] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *CVPR*, 2018.

[19] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *ICCV*, pages 9338–9346, 2019.

[20] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[21] Aleksey Staroverov, D. Yudin, I. N. Belkin, Vasily Adeshkin, Yaroslav K. Solomentsev, and Aleksandr I. Panov. Real-time object navigation with deep neural networks and hierarchical reinforcement learning. *IEEE Access*, 8:195608–195621, 2020.

[22] H. Wang, Weining Wang, and Jing Liu. Temporal memory attention for video semantic segmentation. *arXiv preprint arXiv:2102.08643*, 2021.

[23] Erik Wijmans, Abhishek Kadian, Ari S. Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Decentralized distributed ppo: Solving pointgoal navigation. *arXiv preprint arXiv:1806.1911.00357*, 2019.

[24] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable object navigation. *ICCV*, 2021.

[25] Hongyi Zhang, Moustapha Cissé, Yann Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[26] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *CVPR*, 2017.