Negative Correlation Learning for VLN Agents

Bachelor Thesis
02.10.2022

Johannes Eschbach-Dymanus
Matrikel Nr. 3363713
eschbach@cl.uni-heidelberg.de

Institut für Computerlinguistik
Ruprecht-Karls-Universität Heidelberg

Supervisor: Prof. Dr. Stefan Riezler
1. Reviewer: Prof. Dr. Stefan Riezler
2. Reviewer: Prof. Dr. Michael Herweg

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe verfasst habe und dass alle wörtlich oder sinngemäß aus Veröffentlichungen entnommenen Stellen dieser Arbeit unter Quellenangabe einzeln kenntlich gemacht sind.

Heidelberg, 02.10.2022 _____

Zusammenfassung

Bei Vision and Language Navigation (VLN) wird ein Agent mit natürlichsprachlichen Anweisungen ausgestattet, um durch eine visuelle Umgebung zu navigieren. Jüngste Arbeiten auf diesem Gebiet konzentrieren sich auf die Navigation durch reale städtische Umgebungen mit Straßengrundrissen und entsprechenden 360°-Panoramabildern (Schumann and Riezler, 2022; Chen et al., 2018b). Die Agent-Modelle können hierbei durch Ensembling deutlich bessere Performanz erzielen. Das Ziel dieser Arbeit ist es, die Rolle der Diversität der Modelle in solchen Ensembles zu untersuchen und zu prüfen, ob wir sie explizit ausnutzen können, um die Ensembles zu verbessern. Neben der allgemeinen Performanz wird auch untersucht, ob Agenten-Ensembles im Stande sind vorherzusagen, ob eine von einem *graph-to-text* Modell (Schumann and Riezler, 2020) generierte synthetische Route von einem menschlichen Navigator erfolgreich abgeschlossen werden könnte. Wie bei der reinen Navigationsaufgabe ist auch hier der Effekt der Modelldiversität von speziellem Interesse.

Als theoretische Grundlage für diese Arbeit dient Generalized Negative Correlation Learning (Buschjäger et al., 2020). Hierbei wird der Loss eines Ensembles ℓ in den Durchschnittlichen Loss der Modelle des Ensembles $h_1, \ldots, h_M$ und die Diversität zwischen diesen Modellen zerlegt:

$$\begin{aligned}\ell\left(f(x), y\right) = &\frac{1}{M}\sum_{i=1}^{M}\ell\left(h_i(x), y\right) \\ &- \frac{1}{2M}\sum_{i=1}^{M}\left(h_i(x) - f(x)\right)^T\left(\nabla^2_{f(x)}\ell\left(f\left(x\right), y\right)\right)\left(h_i(x) - f(x)\right)\end{aligned}$$

Basierend auf dieser Zerlegung des Losses und einer Formulierung eines Upper Bounds wird folgende Objective Function für das Training des Ensembles abgeleitet.

$$\frac{1}{N}\sum_{j=1}^{N}\left(\lambda\ell\left(f(x_j), y_j\right) + \frac{1-\lambda}{M}\sum_{i=1}^{M}\ell\left(h_i(x_j), y_j\right)\right)$$

Indem λ zwischen 0 und 1 gesetzt wird, kann beliebig zwischen den beiden Termen interpoliert werden, wobei bei $\lambda = 0$ die Modelle des Ensembles unabhängig von einander trainiert werden. Je höher λ gesetzt wird, desto mehr wird Diversität unter den Teilmodellen des Ensembles 'antrainiert'.

Modell Diversität in Ensembles ist nicht in jedem Szenario oder für jede Art von Modell gleichermaßen vorteilhaft. Diese Arbeit zeigt, dass in einem Vision and Language Navigation Setting eine erhöhte Diversität keinen Vorteil darstellt. Insbesondere werden folgende Schlussfolgerungen gezogen:

1. Diversität auf den Trainingsdaten wirkt sich nur dann positiv auf den Generalisierungsfehler auf ungesehenen Daten aus, wenn das Modell als ganzes einen schlechten Trainingsfit hat. Je besser der Trainingsfit ist, desto weniger profitiert es von der Diversität.

2. Wenn Diversität durch einen $GNCL$-Lernalgorithmus erzwungen wird, erhöht das Ensemble den Trainingsfit, verliert aber an Generalisierungskraft aufgrund von

Overfitting. Dieser Effekt kann für $\lambda < 0.5$ beobachtet werden, wobei bei $\lambda = 0$ die Basismodelle unabhängig voneinander trainiert werden und bei $\lambda = 1$ das Ensemble in einer End-to-End Weise trainiert wird. Für $\lambda = 1$ wird jedoch wiederum ein hoher Loss auf den Trainingsdaten gemessen. Mit mehr GPU-Ressourcen könnte man den gesamten Wertebereich von λ abdecken und untersuchen, ob es ein $\lambda > 0,5$ gibt bei dem hohe Diversität, aber weniger Overfitting, vorliegt.

3. Bei Vision and Language Navigation ist es für ein Ensemble wertvoller, ähnliche statt diverse Agenten zu haben. Die leistungsstärksten Ensembles sind aus Modellen konstruiert, die eine geringe Diversität aufweisen. Ein Ensemble sollte eher auf einer Teilmenge von Routen mit hoher Konfidenz (durch die Übereinstimmung seiner Basismodelle) eine fehlerfreie Leistung erbringen, als auf der gesamten Menge von Routen nur eine einigermaßen gute Leistung. Der letztere Fall führt unweigerlich zu schlechteren VLN-Evaluationsmetriken.

Die Experimente wurden nicht nur durchgeführt, um den Effekt der Diversität zu untersuchen, sondern auch um zu testen, welche Ensemblemethode optimal geeignet ist menschliches Verständnis von synthetischen Navigationsanweisungen vorherzusagen. Für diese zweite Aufgabe wurden folgenden Schlussfolgerungen gemacht:

1. Während das *GNCL* ensemble mit $\lambda = 0.05$ gut abschneidet, ist es nur geringfügig besser als ein Ensemble mit zufällig gewählten Modellen. Die empirische Analyse der Basis-Ensembles zeigte auch keinen positiven Effekt der Diversität der Modelle des Ensembles.

2. Die Task Completion der Ensembles auf Modell-generierten Navigationsinstruktionen korreliert mit der Korrelation zwischen Ensemble und Mensch. Desto erfolgreicher ein Ensemble auf den Routen abschneidet, desto besser kann es menschlichen Erfolg und Misserfolg auf Routen vorhersagen. Unabhängig von der Task Completion kann jedoch einer von vier Erfolgen des Ensembles nicht von Menschen repliziert werden. Selbst wenn man die Task Completion des Ensembles weiter verbessern könnte, würde die Vorhersage-Accuracy daher niemals 82% übersteigen.

3. Diese Obergrenze der erreichbaren Accuracy lässt sich durch Bias in den Trainingsdaten erklären. Da die Ensembles auf denselben Daten wie das Navigationsinstruktionen generierende *graph-to-text* Modell trainiert werden, kann es sich Merkmale merken, die für menschliche Annotatoren unverständlich oder unsichtbar sind. Da das *graph-to-text* Modell ebenfalls diese Merkmale in die generierten Navigationsinstruktionen einbaut, kann das Agenten-Ensemble erfolgreich sein, wo Menschen versagen.

Trotz dieser Beobachtungen waren die Ensembles bei der Vorhersage des Erfolgs menschlicher Annotatoren auf synthetischen Routen deutlich besser als die Majority-Baseline. In einer Folgearbeit sollte untersucht werden, wie weit die Performanz des *graph-to-text* Modells verbessert werden könnte, wenn ein Agenten-Ensemble verwendet wird, um die Decoding-Beams nach ihrer Wahrscheinlichkeit zu ordnen, dass sie von menschlichen Annotatoren abgeschlossen werden können. Zwar kann dise Arbeit keine bevorzugte Ensemble-Methode für eine solche Aufgabe empfehlen, allerdings wurde zumindest festgestellt, dass die Diversität der Modelle des Ensembles bei dieser Wahl nicht berücksichtigt werden sollte.

Contents

1 Introduction

In vision and language navigation (VLN) an agent provided with natural language instructions is tasked to navigate through a visual environment. Recent work in this field focuses on navigation through real-world urban environments with street layouts and corresponding 360° panorama images (Schumann and Riezler, 2022; Chen et al., 2018b). In preliminary experiments[1], Schumann and Riezler (2022) have reported promising results when ensembling VLN agents, greatly increasing their performance. The goal of my work is to investigate the role of model diversity within such ensembles and whether we can explicitly exploit it to improve the ensemble. Aside of the navigation performance, I also investigate whether we can use the ensembles to predict whether a synthetic route generated by a *graph-to-text* model (Schumann and Riezler, 2020) can be successfully completed by a human navigator. As with the pure navigation task, the effect of model diversity is of interest to me.

The prediction diversity of the models in an ensemble is regarded as the main contributor to the degree the ensemble as a whole can reduce generalization error. Buschjäger et al. (2020) establish the theoretical framework of 'Generalized Negative Correlation Learning' (GNCL), in which they describe how the ensemble's loss can be decomposed into the average loss of the individual models and the amount that this loss is reduced by through the models' diversity. Based on the decomposition, they propose an objective function that lets us actively enforce model diversity when training the ensemble jointly. I compare this approach with more naive ensembling methods and analyse the consequences of the joint training in depth.

The experiments conducted in this thesis show that a VLN agent ensemble does not benefit from agent diversity. Instead, an ensemble performs best when base agents show agreement, as it likely allows the ensemble to reliably complete some subset of routes. The ensemble performance deteriorates even more when diversity is enforced through the GNCL learning objective as it facilitates overfitting. This thesis further shows that agent ensembles can be successfully used to classify whether a synthetic, model generated route can be completed by humans or not. While the experiments cannot identify an apparent best ensemble method for this task, they do show that agent diversity has no explanatory power for the classification accuracy of the ensemble and therefore does not need to be taken into account.

2 Task

2.1 VLN-Agent Task

A VLN-Agent is an agent that is tasked to navigate through a visual environment based on natural language navigation instructions. It is important to emphasise the difference between typical instructions a navigation device would provide to a vehicle drivers and

[1] The preliminary experiments are not contained in the 2022 paper. I was made aware of the experiments' results in a private interchange with R. Schumann.

natural language navigation instructions. Natural language instructions are instructions one would provide when asked for assistance on the street. Such instructions are usually based on visible landmarks rather than distances. Tom and Denis (2004) show that instructions based on landmarks are easier memorized by humans than instructions based on distances such as 'turn left in 120 meters'. An excerpt of such human-preferred landmark based instructions could be the following:

> "...Go to the light and turn right at Dunkin Donuts. Follow the next light where Bubble Tea is on the right and turn left..."

As the example demonstrates, the agent needs to combine and understand both directional instructions as well as visual information, in order to successfully follow the instructions.

This first task is identical to the one non-ensembled agents were subjected to in Schumann and Riezler (2022): The agent is required to navigate a discrete environment, a directed graph where each vertex $v \in \mathbb{V}$ is equipped with a 360 degree panorama image p. At any point, the agent is placed on a vertex v and faces some edge $(v, u) \in \mathbb{E}$, where $\mathbb{E}$ is the set of edges of the graph. Since the graph is directed, the edge (v, u) is practically sufficient to describe the agent state. The panorama image shown to the agent needs to be always rotated so that in some state (v, u) the agent would look in the direction of $u \in \mathbb{E}$. To do so, we annotate each edge with an angle $\alpha_{(v,u)}$. Based on the navigation instructions provided, the agent chooses at each time-step t one of four actions $a \in \{\text{FORWARD}, \text{RIGHT}, \text{LEFT}, \text{STOP}\}$ given the state (v, u), where

1. $(a = \text{FORWARD}|(v, u)) \rightarrow (u, w)$, where (u, w) is the edge with an angle $\alpha_{(u,w)}$ most similar to $\alpha_{(v,u)}$.
2. $(a = \text{RIGHT}|(v, u)) \rightarrow (v, w)$, where (u, w) is the next outgoing edge of v when rotating clockwise from angle $\alpha_{(v,u)}$.
3. $(a = \text{LEFT}|(v, u)) \rightarrow (v, w)$, where (u, w) is the next outgoing edge of v when rotating counter-clockwise from angle $\alpha_{(v,u)}$.
4. $(a = \text{STOP}|(v, u))$ stops the agent, making v the terminal node.

If the terminal node is the target node of the navigation task or a direct neighbour of that node, we consider our agent as successful. We measure our agent performance in task completion rate (TC), the percentage of navigation tasks successfully completed.

In Vision Language Navigation, we usually distinct between a 'seen' setting, where the agent is trained and tested within the same environment and an 'unseen' setting, where the test routes are situated in an unknown environment. In this thesis, we evaluate the agent ensemble performance for the 'seen' setting. Although the test routes are routes the agents never traversed before, it likely will have visited each node already as part of another route during training. Nonetheless, the agent will have to generalize, as it will not necessarily have to take the same action at the node in question. Furthermore, the navigation instructions will not necessarily make use of the same landmark for orientation.

2.2 System Generated Route Quality Estimation

While the first task mostly serves to compare GNCL with other ensembling methods, the second task will also make a contribution to natural language navigation research. In this task, we use the trained agent-ensembles to predict whether a human could successfully complete synthetic, system generated instructions.

The synthetic instructions are generated by the *graph-to-text* model trained by Schumann and Riezler (2020). The model receives an OpenStreetMap[2] representation of the route and is trained to produce natural language navigation instructions. In their paper, Schumann and Riezler (2020) record that only 55% of the generated routes can be successfully completed by humans in a Google Street View virtual environment.

Since the VLN agent can be trained to navigate the same Street View environment, the next logical step is to see whether the agent can distinguish good system generated instructions from bad ones. If we train the agent on instructions compiled and validated by human annotators, it should more likely succeed on routes with instructions similar to those a human would give. If the agent indeed fails where humans would fail and succeeds where they would succeed, we could use it to re-rank the decoding beams of the instruction generating model.

While we lack access to annotators in order to conduct the beam reranking and measure the performance boost, we will instead use the correlation between human annotator successes on the test set from Schumann and Riezler (2020) and VLN agent successes on the same routes as a proxy for the viability of the endeavour. We treat this as a binary classification problem, where navigation success and failure are the two classes and the human successes and failures are the gold standards. Treating the agent successes and failures respectively as the predictions, we can evaluate our agent's performance through accuracy. By framing the task as a classification problem, we can furthermore make use of the confusion matrix and deduct conclusions from the precision and recall metrics of the success and failure class. As in the first task, we will compare the GNCL ensemble with different ensemble methods and investigate whether there is any benefit to it.

3 Data

We use the *map2seq* dataset compiled by Schumann and Riezler (2020) for our experiments. The *map2seq* environment is situated in Manhattan and consists of 29,641 nodes, each equipped with a 360° panorama image. In total, the dataset contains 7,672 navigation instructions whose corresponding routes are between 35 and 45 nodes long and always contain the shortest path from start to goal location. The navigation instructions are on average 55.1 tokens long. Since the model in Schumann and Riezler (2020) was tasked to generate instructions based on OpenStreetMap data, the human annotators compiling the navigation instructions were provided only with the map segment displaying the route and landmarks such as restaurants or museums. The navigation instructions were then validated by having two other human annotators follow the instructions in a static Google StreetView environment compiled by Chen et al. (2018a)

[2]www.openstreetmap.org

and controlling whether they are able to do so. If one of them succeeds, the route is considered valid, navigate-able. This validation was necessary, as certain landmarks on OpenStreetMap might not be visible in the StreetView environment. If instructions with such landmarks were included in the training data, the *graph-to-text* model would have learned to make use of the landmarks. During evaluation, it would have been unfairly punished as, unsurprisingly, humans could not follow generated instructions containing such landmarks.

In order to conduct the agent-annotator correlation task, we need to train our VLN agents on data that does not intersect with the test routes the *graph-to-text* model generated the navigation instructions for. In their paper, Schumann and Riezler (2020) evaluated the *graph-to-text* model in a 'partially seen' and an 'unseen' environment with 700 routes each. In this thesis we will make use of the 'partially seen'[3] test set. Consequently, we will also use those same routes, but with corresponding human instructions instead of the system generated ones, as test set in the VLN-agent navigation task. This way, we only need to train the models once and can evaluate them on both tasks. After isolating those 700 routes, the remaining routes were split into a train set of 6293 and a validation set of 622 routes.

4 Model Architecture

While the experiments involve two types of models, the *graph-to-text* model and the VLN agent model, the former one is only involved in the agent-annotator correlation task and even then the precise architecture is not important. The research interest lays in whether the VLN agents can effectively discriminate between correct, human understandable instructions and incorrect instructions. How exactly theses system generated instructions were decoded is irrelevant for the task at hand. The architecture of the VLN agents we ensemble, however, deserves some attention as the design choices could hold explanations to some of the observations made throughout this thesis. The agent architecture and the code implementation[4] is directly adopted from Schumann and Riezler (2022), who in turn were inspired by the cross-modal attention model by Krantz et al. (2020). Overall, the agent architecture consists of three parts, an instruction encoder, a visual encoder and an LSTM decoder recurrently attending the two encoders.

The instruction encoder is a bidirectional LSTM (Graves et al., 2005) that encodes the embedding of each token x_1 to x_L into the hidden states w_1 to w_L. The last cell state z_L^w of the encoder LSTM is used to initialize the LSTM decoder.

The visual encoder rotates the respective 360° panorama image p so that it aligns with the agent's heading direction $\alpha_{(v,u)}$ (at agent state (v, u)). The image is then sliced into 6 equal parts, of which all but the 60° segment 'behind' the agent are fed into a ResNet-50 [5] pretrained on ImageNet (Russakovsky et al., 2014). For each of the five slices, the pre-final ResNet layer is extracted, leaving us with the five final vector representations

[3]There is no distinction between the 'partially seen' and the 'seen' setting. The word 'partially' is used to clarify that not necessarily every single node within the test set routes was already covered by a training route. The same, however, holds true to the 'seen' setting in **?**

[4]https://github.com/raphael-sch/map2seq$_v ln$

[5]https://pytorch.org/vision/0.8/models.html

$\bar{p}^1$ to $\bar{p}^5$ which are then concatenated and passed through a feed forward neural network to provide us with the final visual encoding $\bar{p}$. While Schumann and Riezler (2022) also experimented with other configurations, the pre-final ResNet-50 layer yielded best results for the 'seen' settings and therefore is the one used in this thesis.

Aside of the visual encoding p_t at time step t, the first LSTM decoder layer also receives a previous action embedding $\bar{a}_{t-1}$, a junction type embedding $\bar{n}_t$ and a heading delta d_t; all concatenated ($\oplus$) together. The junction type embedding contains information about the number of outgoing edges at the current node, while the heading delta signals any rotation the agent was subjected to by Google Street View. In total, the output of the first decoder layer is:

$$h_t^{\text{first}} = LSTM^{\text{first}}\left([\bar{a}_{t-1} \oplus \bar{n}_t \oplus d_t \oplus \bar{p}_t]\right)$$

This output is then attending the hidden states of the instruction encoder to receive c_t^w, which in turn then attends the sliced visual representations:

$$\begin{aligned} c_t^w &= \text{MultiHeadAttention}(h_t^{\text{first}}, (w_1, \ldots, w_L)) \\ c_t^p &= \text{MultiHeadAttention}(c_t^w, \bar{p}^1, \ldots, \bar{p}^5) \end{aligned}$$

The contextualised representations are then concatenated with the output h_t^{first} of the decoder's first layer and an embedding $\bar{t}$ of time step t.

$$h_t^{\text{second}} = LSTM^{\text{second}}\left(\left[\bar{t} \oplus h_t^{\text{first}} \oplus c_t^w \oplus c_t^p\right]\right)$$

The representation h_t^{second} is then passed through a feed forward network to get the final output vector $\bar{a}_t$ = containing the prediction scores of the four actions {FORWARD, RIGHT, LEFT, STOP}.

5 Related Work - Ensembling & Model Diversity

In general, ensembling refers to the process of combining multiple models in a prediction process. While there are numerous ensemble methods, the most basic and straightforward one is to simply average the predictions of individual models. For any convex loss function ℓ, we can show with help of the Jensen's inequality that the expected loss of the averaged outputs of M models is always lower than the average of the expected loss of the M individual models. Suppose that each model makes an error ϵ_i on each data point, we get

$$\begin{aligned} \mathbb{E}\left[\ell\left(\frac{1}{M}\sum_i^M \epsilon_i\right)\right] &\leq \mathbb{E}\left[\frac{1}{M}\sum_i^M \ell\left(\epsilon_i\right)\right] \\ \ell\left(\frac{1}{M}\sum_i^M \mathbb{E}\left[\epsilon_i\right]\right) &\leq \frac{1}{M}\sum_i^M \ell\left(\mathbb{E}\left[\epsilon_i\right]\right) \end{aligned} \tag{1}$$

Loss or error can be decomposed into the contributions of model bias and variance. Perrone et al. (1994) show for the Mean Square Error (MSE) that the overall loss reduction through model averaging must come exclusively from a reduction in variance. Consider we sample models h_1 to h_M from some model distribution $\mathcal{H}$ to create our ensemble

f. Since the bias contribution to the MSE equals $\mathbb{E}[(\mathbb{E}[f] - f^*(x))^2]$, where f^* is the target function, and $\mathbb{E}[f] = \mathbb{E}[\frac{1}{M}\sum_i^M h_i] = \frac{1}{M}\sum_i^M \mathbb{E}[h_i] = \mathbb{E}[h_i]$, we know that the bias remains unchanged. Consequently, averaging the models must reduce the overall prediction variance to decrease the total MSE.

For the expected squared error, Goodfellow et al. (2016) offer insights into what formally causes the loss reduction. Presume that the errors ϵ_i are drawn from a zero-mean multivariate normal distribution with variances $\mathbb{E}[\epsilon_i^2] = v$ and pairwise co-variances $\mathbb{E}[\epsilon_i \epsilon_j] = c$. We can now decompose the ensemble mean square error into variance and covariance of the M models' predictions.

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{1}{M}\sum_i^M \epsilon_i\right)^2\right] &= \frac{1}{M^2}\mathbb{E}\left[\sum_i^M \left(\epsilon_i^2 + \sum_{j\neq i}^M \epsilon_i \epsilon_j\right)\right] \\
&= \frac{1}{M^2}\sum_i^M \mathbb{E}\left[\epsilon_i^2\right] + \frac{1}{M^2}\sum_i^M \sum_{j\neq i}^M \mathbb{E}\left[\epsilon_i \epsilon_j\right] \\
&= \frac{1}{M}v + \frac{M-1}{M}c
\end{aligned} \tag{2}$$

The smaller the correlation between the models' errors, the more the entire term shrinks to $\frac{1}{M}v$ and the more the entire terms becomes inversely proportional to the number of models in the ensemble. Goodfellow et al. (2016) demonstrates with this the beneficial effect of prediction diversity among the models used in the ensemble. The more diverse the models, the less their errors correlate, the smaller the overall squared error.

While it is difficult to trace back the first research paper discussing the positive relationship between model prediction diversity and ensemble loss, we can certainly assert that it has been a constant research interest throughout the last decades (Perrone and Cooper, 1993; Kuncheva and Whitaker, 2003; Brown and Kuncheva, 2010; Melville and Mooney, 2005; Liu and Yao, 1999). Perrone and Cooper (1993), for instance, developed an ensemble method that allows us to incrementally build a weighted average ensemble based on the individual model MSEs and the prediction covariance matrix between the models' errors. At any point, it is only then beneficial to add the next best model to the ensemble if the covariance is lower than some computable upper bound.

The relation between model diversity and ensemble performance, however, is not always that straight forward. This is especially the case for classification tasks, where the easy to decompose MSE cannot be used effectively. Kuncheva and Whitaker (2003) analyse the correlation of various model diversity metrics with the ensemble's accuracy in classification tasks and come to mixed conclusions. They conclude that in most real-life pattern recognition problems, diversity measures are not useful when building classifier ensembles. Brown and Kuncheva (2010) show that if we do not average the model predictions but instead use a voting system and a zero-one loss, however, we can once again formally decompose the loss and isolate the contribution of model diversity. Interestingly, they observe that only the diversity on the correctly predicted data points benefits the ensemble, while diversity on incorrectly predictions is harmful.

The recognition of the beneficial effect of diversity also gave rise to a number of negative-correlation-learning (NCL) algorithms which aim to enforce diversity already during model training (Liu and Yao, 1999; Brown et al., 2005; Opitz et al., 2017; Webb et al.,

2019). To do so, the models are trained jointly in parallel so that a regularizer term that encourages diversity can be added to each prediction's loss calculation. While previous NCL papers investigated special cases such as specific losses, Buschjäger et al. (2020) present a generalized formulation and theoretical framework that encompasses all other NCL algorithms.

6 Generalized Negative Correlation Learning

In this section I will reiterate the mathematical derivation of the Generalized Bias-Variance Decomposition by Buschjäger et al. (2020). As they published their paper in an 8 pages journal article, their derivation is quite dense, subscripts are dropped and verbal explanations are kept at a minimum. With a more lengthy and purpose driven account I hope to provide a didactically better approach to the theory at hand. To further avoid redundancy, I will exemplify the theory directly with the VLN task at hand rather than repeating it in an abstract, general manner.

In our example, we have labeled training data $\mathcal{S} = \{(x_j, y_j) \mid i = j, \dots, N\}$ i.i.d. sampled from a data distribution $\mathcal{D}$ where x_j consists of textual instructions and a start heading and y_i is a sequence of actions a_1 to a_T. Since we use teacher forcing during training, it is more convenient to describe a single data point as the features and target for one single prediction. In this case, the feature vector $x_j \in \mathcal{X} \subseteq \mathbb{R}^d$ consists the entirety of input features (the LSTM hidden- and cell state, instruction encoding, current visual encoding etc.) collected by the teacher-enforced actions leading up to the data point in questions and the target $y_j \in \mathbb{R}^C$ is a one-hot encoding of the desired action out of the $C = 4$ options {FORWARD, RIGHT, LEFT, STOP}.

The VLN agent is a model h that maps the input space $\mathcal{X}$ to scores over the actions in $\mathbb{R}^C$ and belongs to the model class $\mathcal{H} = \{h : \mathcal{X} \to \mathbb{R}^C\}$ of all possible VLN agents. Since model fitting algorithms like Stochastic Gradient Descent involves some randomization, models fit to our training data $\mathcal{S}$ will follow some distribution Θ over $\mathcal{H}$.

6.1 A Generalized Bias-Variance Decomposition

Given a large enough ensemble of M models sampled from Θ, we can approximate the loss of the distribution's expected model

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\mathbb{E}_{h\sim\Theta}\left[h(x)\right]\right)\right] \approx \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\frac{1}{M}\sum_{i=1}^{M} h_i(x)\right)\right] \quad \text{, where } h_i \sim \Theta \tag{3}$$

Ultimately, we aim to identify how model diversity contributes to this term to establish the theoretical foundation for GNCL. Knowing that the base models of an ensemble approximate the distribution Θ we go now ahead and decompose the expected loss of a fit agent

$$\mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\ell\left(h\left(x\right), y\right)\right] \tag{4}$$

into what can be explained by the bias and variance (or diversity) of the model distribution Θ. Keep in mind that equation 3 is the expected loss (over $\mathcal{D}$) of the expected

model (over Θ), while equation 4 is the expected loss with expectation over both $\mathcal{D}$ and Θ. Since the loss function is not linear, the expected loss of a model is not equal to the loss of the expected model. Ultimately, we want to isolate the term in 3 as it represents our ensemble. To get there, we will approximate term 4 and then rearrange the resulting equation.

The following approximation requires the loss function ℓ to be twice differentiable, which is true for the Cross Entropy Loss employed in this thesis as well as for most other loss functions used in deep learning. Note also that we are looking here at the expected loss of the agent distribution Θ on the data distribution $\mathcal{D}$. To clarify, the distribution $\mathcal{D}$ is not the training data $\mathcal{S}$ the agent is fit on, but the distribution of data both training and test data are sampled from.

This derivation is not a loss decomposition in the classical sense, where the bias is the error between the agent's expectation (in regards to $\mathcal{D}$) and the real expected value and the variance is the jumpiness of the agent's predictions around their mean $\frac{1}{|\mathcal{D}|}\sum_{x\sim\mathcal{D}} h(x)$. Instead, given the distribution of agents Θ, we decompose the loss into the expected loss of the expected agent μ

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\mu(x), y\right)\right], \quad \text{where } \mu(x) = \mathbb{E}_{h\sim\Theta}\left[h(x)\right] \tag{5}$$

and the expected variance of agent predictions around their expected prediction $\mu(x)$ for any $x \sim \mathcal{D}$

$$\mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\left(h(x)-\mu(x)\right)^T\left(h(x)-\mu(x)\right)\right]. \tag{6}$$

Note that term 6 does not denote the variance of the predictions over $\mathcal{D}$. Instead, it measures how much the agent predictions to vary at each $x \sim \mathcal{D}$ throughout $\mathcal{D}$.

In order to decompose the expected loss of a fit agent 4 into the quantities described by terms 5 and 6, we can use a second-order Taylor approximation of the loss ℓ around the expected agent μ:

$$\begin{aligned}
\mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\ell\left(h\left(x\right), y\right)\right] = \; & \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\mu(x), y\right)\right] \\
& + \mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\left(h(x)-\mu(x)\right)^T\nabla_\mu\ell\left(\mu\left(x\right), y\right)\right] \\
& + \mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\frac{1}{2}\left(h(x)-\mu(x)\right)^T\nabla_\mu^2\ell\left(\mu\left(x\right), y\right)\left(h(x)-\mu(x)\right)\right] \\
& + \mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[R\right]
\end{aligned} \tag{7}$$

where R is remainder of the Taylor approximation. Given any test point $(x, y) \sim \mathcal{D}$, the prediction and therefore loss of the expected agent μ is a constant in regards to $\mathbb{E}_{h\sim\Theta}$ and therefore $\mathbb{E}_{h\sim\Theta}\left[\nabla_\mu\ell\left(\mu\left(x\right)\right)\right] = \nabla_\mu\ell\left(\mu\left(x\right)\right)$. This allows us to rewrite the second summand as

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{E}_{h\sim\Theta}\left[\left(h(x)-\mu(x)\right)^T\right]\nabla_\mu\ell\left(\mu\left(x\right), y\right)\right] \tag{8}$$

Since by definition $\mu(x) = \mathbb{E}_{h\sim\Theta}\left[h(x)\right]$, the expected difference between an agent's prediction and the expected agent's prediction $\mathbb{E}_{h\sim\Theta}\left[\left(h(x)-\mu(x)\right)^T\right]$ will zero out, which in turn makes the entire term 8 zero out.

In appendix 11.1.1 we demonstrate that the remainder $\mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}[R]$ is bounded for the Cross Entropy Loss employed in this thesis[6]. With the second summand zeroing out and a small enough remainder, we are left with following approximation:

$$\begin{aligned}\mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\ell\left(h\left(x\right),y\right)\right] &= \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\mu(x),y\right)\right] \\ &+ \mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\frac{1}{2}\left(h(x)-\mu(x)\right)^T\nabla_\mu^2\ell\left(\mu\left(x\right),y\right)\left(h(x)-\mu(x)\right)\right]\end{aligned} \tag{9}$$

The expected loss of an agent h is now decomposed into the two desired quantities. The first summand is the expected loss of the expected agents (eq. 5) and so to say the bias of the distribution of fit agents Θ. The second summand expresses the variance of the predictions of models stemming from Θ and matches our desired quantity (eq. 6) up to the loss specific factor $\nabla_\mu^2\ell\left(\mu\left(x\right),y\right)$. As this factor is not dependent on h it is a constant when we calculate the expected loss over all $(x,y)\sim\mathcal{D}$.Buschjäger et al. (2020) state that the second summand can be interpreted as the 'co-variance of h with respect to the expected model μ'. While technically correct, I consider this phrasing prone to misunderstanding and want to therefore clarify that the variance term 6 is inverse to the covariance of the predictions of agents sampled from Θ on the data $\mathcal{D}$. If the model predictions are expected to vary greatly **in regards to their mean** at any data point $(x,y)\sim\mathcal{D}$, they have a low co-variance over the entire data $\mathcal{D}$ and vice versa.

For convex loss functions the second derivative $\nabla_\mu^2\ell\left(\mu\left(x\right),y\right)$ is positive definite. Therefore, the expected loss of an agent increases with increased variance between the predictions of the agents of Θ. If we rearrange equation 9, however, we see that at constant expected loss of agents, increased agent diversity results in decreased expected loss of the expected agent.

$$\begin{aligned}\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell\left(\mu(x),y\right)\right] &= \mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\ell\left(h\left(x\right),y\right)\right] \\ &- \mathbb{E}_{h\sim\Theta,(x,y)\sim\mathcal{D}}\left[\frac{1}{2}\left(h(x)-\mu(x)\right)^T\nabla_\mu^2\ell\left(\mu\left(x\right),y\right)\left(h(x)-\mu(x)\right)\right]\end{aligned} \tag{10}$$

We can now transfer this realization to the concrete usage of ensembles. If an ensembled agent f is sufficiently large, its base agents h_1 to h_M approximate the distribution Θ. As stated in 3 its prediction $f(x)=\frac{1}{M}\sum_{i=1}^M h_i(x)$ therefore approximates the prediction of the expected agent $\mu(x)$ and the base agents' variance term approximates the one of the distribution. Consequently we, can decompose the loss of such an ensemble f as follows:

$$\begin{aligned}\ell\left(f(x),y\right) &= \frac{1}{M}\sum_{i=1}^M\ell\left(h_i(x),y\right) \\ &- \frac{1}{2M}\sum_{i=1}^M\left(h_i(x)-f(x)\right)^T\left(\nabla_{f(x)}^2\ell\left(f\left(x\right),y\right)\right)\left(h_i(x)-f(x)\right)\end{aligned} \tag{11}$$

With the second derivative of a convex loss function being positive definite, Buschjäger et al. (2020) conclude that an ensemble of agents is therefore always better than a single agent. We can also record that given two agent ensembles with equal good base agents, the one with more diverse base agents will obtain the lower loss.

[6]The derivation by Buschjäger et al. (2020) unfortunately contains faulty indices, a variable mix-up and a minor calculation mistake ($\frac{2}{3^3}\neq\frac{1}{27}$). For readability, I therefore included a more tidy version in the appendix.

6.2 GNCL Objective

Just as 4.2, this section is a reiteration of Buschjäger et al. (2020) and is not my own research.

If we train an ensemble of agents jointly, we can use equation 11 as learning objective. This allows us to explicitly regulate by how much we want to reduce the ensemble loss by *a.* reducing the base agents' losses directly, or *b.* increasing base agent prediction diversity. To do so, we simply need to add a hyperparameter λ:

$$\begin{aligned}\ell\left(f(x), y\right) = &\frac{1}{M}\sum_{i=1}^{M}\ell\left(h_i(x), y\right) \\ &- \frac{\lambda}{2M}\sum_{i=1}^{M}\left(h_i(x) - f(x)\right)^T\left(\nabla^2_{f(x)}\ell\left(f\left(x\right), y\right)\right)\left(h_i(x) - f(x)\right)\end{aligned} \tag{12}$$

Given base agents h_1 to h_M powerful enough so that they can achieve perfect zero training loss, we could simply set $\lambda = 0$ to ignore the second term of the objective, since it would by necessity become zero as well. This, however, would mean that all base agents make the same predictions for each data point, which would defeat the whole point of an ensemble. As discussed in 4.2, we want to ensemble diverse base learners to achieve better generalization error on unseen data. Therefore, we purposefully choose $\lambda > 0$ to achieve this desired effect.

To further clarify this point made by Buschjäger et al. (2020), I want add that the loss decomposition in 11 naturally also applies to any arbitrary data, not just to the one the base learners were trained on. Consider two ensembles f_1 and f_2 with near zero training loss, where the former has optimal base learners and the latter has less optimal, but diverse base learners. If we now have the ensembles predict unseen data which, by definition, differs from the training data, the base learners of f_1 will not carry over their perfect zero loss and the ensemble will suffer generalization error. Although the base learners of f_2 will also suffer generalization error, the diversity term in 11 will remain constant and therefore reduce the loss. This explanation is obviously an oversimplification and does not account for a number of quantities. For instance, f_1's better base learners will likely fit the unseen data better than f_2's base learners and thereby possibly compensate for the smaller diversity term.

The learning objective in 11 comes with some serious difficulties, though. First of all, the computation of $\nabla^2_{f(x)}\ell\left(f\left(x\right), y\right)$ can be costly. For the Cross-Entropy-Loss the computation involves both an otherwise unnecessary Softmax calculation as well as multiple matrix multiplications[7]. Secondly, we currently assume that the remainder $\mathcal{R}$ in 7 is small enough to neglect it. This, however, cant be guaranteed, as the upper bound of the remainder for the Cross Entropy Loss might be small, but not entirely irrelevant. Considering that we can fit parameter rich Neural Network like our agents to training data quite well, we cannot say for certain how large this quantity really is in relation to the ensemble loss. In fact, section 9.1.1 confirms that on training data, the decomposition error is magnitudes larger than the ensemble loss, which ultimately renders this learning

[7]The implementation of the GNCL learning algorithm is based on a sample implementation by Buschjäger et al. (2020); see https://github.com/sbuschjaeger/gncl.

objective in its current state unusable. Buschjäger et al. (2020) circumvent this issue by deriving an upper bound for the objective 11:

$$\begin{aligned}
&\frac{1}{M}\sum_{i=1}^{M}\ell\left(h_i(x),y\right)-\frac{1}{2M}\sum_{i=1}^{M}\left(h_i(x)-f(x)\right)^T\left(\nabla^2_{f(x)}\ell\left(f\left(x\right),y\right)\right)\left(h_i(x)-f(x)\right)\\
&\leq \frac{\mathbf{2}}{M}\sum_{i=1}^{M}\ell\left(h_i(x),y\right)-\frac{1}{2M}\sum_{i=1}^{M}\left(h_i(x)-f(x)\right)^T\left(\nabla^2_{f(x)}\ell\left(f\left(x\right),y\right)\right)\left(h_i(x)-f(x)\right) \quad (13)\\
&=\ell\left(f(x),y\right)+\frac{1}{M}\sum_{i=1}^{M}\ell\left(h_i(x),y\right)
\end{aligned}$$

Since the ensemble loss $\ell(f)$ itself contains the diversity term, we can regulate between base learner optimization and diversification by interpolating between the ensemble loss and the average base-learner loss. With a training set of N data points we get following final Generalized Negative Correlation Learning objective:

$$\frac{1}{N}\sum_{j=1}^{N}\left(\lambda\ell\left(f(x_j),y_j\right)+\frac{1-\lambda}{M}\sum_{i=1}^{M}\ell\left(h_i(x_j),y_j\right)\right) \quad (14)$$

7 Empirical Analysis of Theoretical Framework

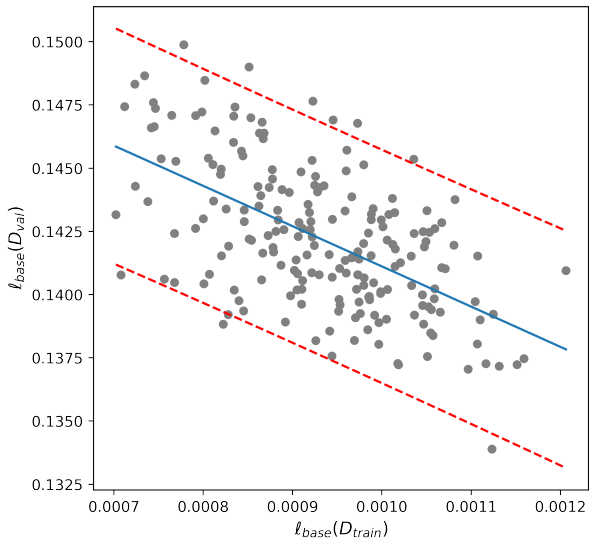
In this chapter I empirically investigate how the different quantities involved in the loss decomposition precisely affect the generalization error of the ensemble. The loss of our ensembles can be decomposed into the average base agent loss, the base agent diversity and the decomposition error. We will refer to these three quantities from here on simply as ℓ_{base}, Div and $\mathcal{R}$. The total ensemble loss will be signified with simply ℓ.

One might wonder why an empirical analysis is necessary with such elaborate mathematical derivations provided. As already briefly discussed in section 4.4.1, we do not know how exactly the quantities carry over from the loss on seen data to the one on unseen data. Particularly, given two ensembles with equal loss ℓ on our training data but different proportions of ℓ_{base} and Div, we want to be able to confidently predict not only which of the two will generalize better to our validation data but also by how much.

In order to investigate the effects, I will make use of Generalized Additive Models (Hastie, 2017). Unlike Linear Models, Generalized Additive Models can capture non-linear relationships between each predictor and the predicted variable. The dependent variable is then predicted by the sum of those non-linear functions modeled for each predictor, which allows us to isolate their effect. I use the pyGAM python library by Servén and Brummitt (2018) to fit the GAMs and plot the results. As non-linear approximator function for the predictors I make use of B-Splines. For measurement data, I trained 25 agents and then random sampled sets of six models to construct 200 ensembles (out of a possible 177100 combinations). An overview over the 25 agents is provided in the appendix in table 3.

7.1 Base Agent Loss

It is intuitive to assume that the better the base agents perform on the training data, the better they will perform on the validation data. While this is certainly true for the most part - after all, we train the agents for a reason - we cannot confidently assume such a correlation to be present among fully converged agents. Figure 1 shows that there is indeed a negative relationship between training and validation loss, suggesting overfitting of the individual agents. This observation is actually promising from a GNCL perspective, as this allow would allow us to trade base agent loss for diversity without any costs but instead with double the benefit.



Spline	p-value	Sig. Code
$\ell_{\text{base}}(D_{\text{train}})$	1.11e−16	***
intercept	1.11e−16	***
Pseudo R-Squared:		0.34

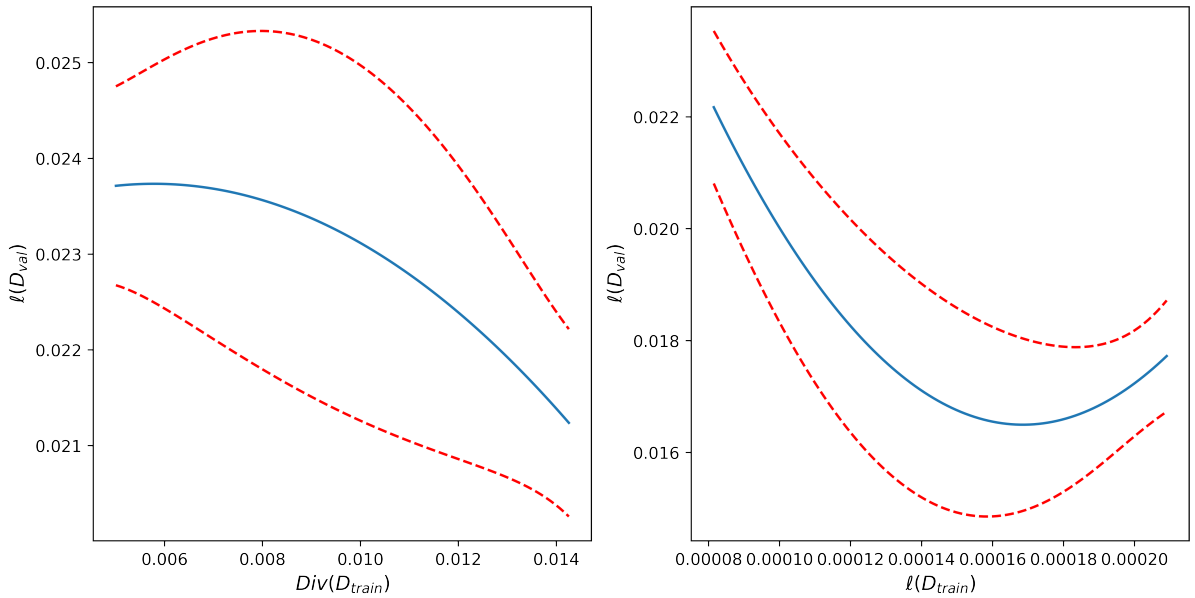
Significance codes: ***: $p < 0.001$, **: $p < 0.01$:, *: $p < 0.05$

Figure 1: Correlation between Base Agent Train and Test Loss

I make use of this example to exemplify how to interpret the GAM result table. The B-Spline function of $\ell_{\text{base}}(D_{\text{train}})$ in figure 1 models the relationship between the base agent loss on the training set $\ell_{\text{base}}(D_{\text{train}})$ and the validation set $\ell_{\text{base}}(D_{\text{train}})$ with statistical significance of $p < 0.001$. The intercept term is also significant on the same level. The dashed red line in the plot depicts the 95% confidence bounds of the spline. Although the type of Pseudo R-Squared used is not specified in the PyGAM documentation, a look at the source code suggests that the R_L^2 by Cohen et al. (2002) is used. For the example at hand this means that the full model of intercept plus splines reduces the deviance by 35.9% compared to the intercept-only model.

7.2 Diversity

There are a number of reasons why an empirical analysis is necessary to identify the effect of the base agents' training data prediction diversity on the generalization error. We do not know to what degree ensembles with higher base agent prediction diversity on training data achieve higher diversity on unseen data as well. For our data, a Pearson's correlation coefficient $r = 0.37$ suggests that there is only a moderate positive correlation between the two quantities.



Spline	p-value	Sig. Code
$Div(D_{\text{train}})$	0.0086	**
$\ell_{\text{base}}(D_{\text{train}})$	1.11e−16	***
intercept	1.11e−16	***
Pseudo R-Squared:	0.31	

Significance codes: ***: $p < 0.001$, **: $p < 0.01$:, *: $p < 0.05$

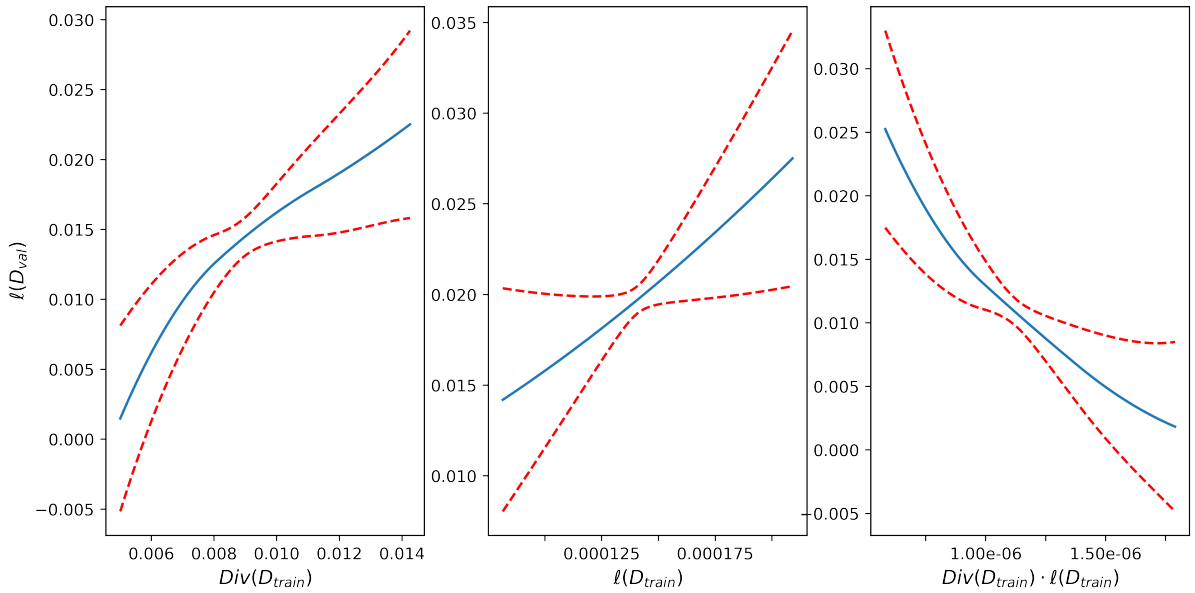
Figure 2: Effect of Diversity on Validation Loss at Constant Total Train Loss

We could now go ahead and isolate the effect of diversity under constant base agent loss. From a GNCL perspective, however, the two quantities actually measure the very same thing, under constant loss. Models with identical total loss could either achieve this loss by having low base agent loss and low diversity, or by having higher base agent loss for which consequently higher diversity is needed to compensate for. To investigate the potential of GNCL, we therefore want to measure the effect of increased diversity at constant training loss of the ensemble.

The right plot in figure 2 shows the contributions to the validation loss of both the training loss and the diversity. The spline function for the training loss displays typical overfitting behaviour; with high statistical significance. Up to a certain point we can diminish validation loss by reducing train loss. Anywhere past this point - here 0.00016 - further loss reduction harms the generalization capacity of the ensemble. The left plot shows the effect of training diversity when keeping training loss constant. As expected, increasing the diversity at constant loss reduces the generalization error. Surprisingly, however, this relationship between the diversity and validation loss is not a linear one. Instead, we see increasing returns with increasing diversity, suggesting that diversity is only worth to invest in if an ensemble does so sufficiently.

While we could view these results as sufficient proof for the potential of GNCL ensembles, we should be wary of interaction effects between diversity and the overall training loss. It might be the case that high diversity is only beneficial at either high or low losses.

To account for this we introduce an interaction term $Div(D_{\text{train}}) \cdot \ell(D_{\text{train}})$ and refit the Generalized Additive Model.



Spline	p-value	Sig. Code
$Div(D_{\text{train}})$	2.79e−5	***
$\ell(D_{\text{train}})$	1.14e−5	***
$Div(D_{\text{train}}) \cdot \ell(D_{\text{train}})$	1.25e−10	***
intercept	1.11e−16	***
Pseudo R-Squared:	0.34	

Significance codes: ***: $p < 0.001$, **: $p < 0.01$:, *: $p < 0.05$

Figure 3: Effect of Diversity on Validation Loss with Interaction Term

Surprisingly, when we include the interaction term, we now observe the opposite relation between training diversity and validation loss (see figure 3). The training loss spline function as well is entirely inverted and shows no signs of ensemble over-fitting anymore. The interaction term, however, holds the answer to these extreme changes. As we can see, higher diversity becomes more beneficial the higher the overall training loss of the ensemble is. This analysis shows that if very low loss can be achieved on training data, it is preferable to do so through low base agent losses and low diversity rather than through high base agent losses and high diversity.

8 Ensemble Overview

8.1 Single

The *Single* model is not an ensemble but an individual agent. This model serves as a reference on how much benefit we get from ensembling with the respective method compared to not ensembling at all. Overall, I trained 25 such agents.

8.2 Baseline

The *Baseline* ensembles consist of six independently trained agents sampled from the 25 agents available. As with all other ensembles in this thesis, they simply average the outputs of their base agents to land a prediction. This ensemble represents the most common but also easiest to implement ensemble method and therefore serves as the baseline. These ensembles are equivalent to the ones analysed in section 7.

8.3 Naive

With 25 agents available, it seems sensible to not just random sample a set of six agents to build the ensemble but instead find the best combination of individual agents. However, with 25 agents available we would have to exhaustively search through the entire power set (minus the empty set) of the agents. As searching through 30 million possible subsets is not a feasible task, we need to find an easier solution on how to identify good ensembles. A naive approximation would be to simply sort the agents by their validation set performance and select the top K to build the ensemble. This exact procedure is used to create the $Naive$ ensemble, with $K = 6$ base agents to keep the ensembles comparable.

The reason I include this procedure is that it can give us feedback on the importance of the base agent losses. By creating the ensemble this way, we completely disregard the benefits of diversity. If the GNCL ensembles are unable to beat this method, we need to conclude that is is more worthwhile to get better base agents than it is to artificially enforce diversity.

8.4 Greedy

A better way to find a good ensemble among the available agents would be to greedy search it. With N agents available and a maximum ensemble size of K, we can describe the search algorithm as follows:

```
Function GreedySearch(agents, K):
    ensemble ← ∅
    best_score ← 0
    for k ← 0 to K do
        best_agent ← ∅
        foreach agent ∈ agents do
            new_ensemble ← ensemble + agent
            val_score ← evaluate(new_ensemble)
            if val_score > best_score then
                best_score ← val_score
                best_agent ← agent
            end
        end
        if best_agent = ∅ then
            break
        end
        agents.remove(best_agent)
        ensemble ← ensemble + best_agent
    end
    return ensemble
```

Algorithm 1: Greedy Construction of Ensemble

This algorithm ensures that we add at each step the agent which benefit the ensemble's validation performance most. Compared to the *Naive* method, this method can choose the agents not only based on their independent loss, but can also purposefully select worse agents if they benefit the ensemble through diversity. The ensemble constructed with this method will likely be the highest performing one. The loss decomposition will allow us to investigate whether the algorithm indeed opted to build an ensemble with high diversity, which in turn provides us with feedback on the importance of diversity.

In this thesis I include two ensembles constructed in such way. The $Greedy_{25}$ ensemble greedy selects up to $K = 6$ agents out of a total of $N = 25$ possible candidates. The $Greedy_6$ ensemble greedy selects up to $K = 6$ agents out of a total of $N = 6$ agents randomly selected from the 25 available ones. The purpose of the second ensemble is to provide better comparability with the GNCL ensembles to allow statements such as 'if we were to train six agents, we should favor method A over B to construct our ensemble'. An overview over the concrete models selected for the $Greedy_{25}$ ensembles is provided in the appendix in table 3.

8.5 GNCL

The *GNCL* ensembles are trained jointly with the objective function 14. The respective λ regularizer chosen is signified by the subscript. For instance, the $GNCL_{0.05}$ ensemble is trained with $\lambda = 0.05$.

9 Evaluation

9.1 VLN Agent Performance

In this section we evaluate the various ensembles' performances on the *map2seq* data-set. We measure performance in task completion (TC), the percentage of routes where the agent stopped on the target node or a direct neighbour of it. Checkpoint and model selection is based on the Shortest Path Distance (SPD) on the validation set. The SPD measures the number of nodes between the target node and the one the agent has stopped on. We use this metric for checkpoint and model selection because it provides a more reliable estimate of the generalization error of the ensemble than the binary TC does.

To ensure reliability of the results, I trained each ensemble with five different random seeds and then averaged the evaluation metrics. It would certainly be desirable to have more repetitions in order to narrow confidence bounds, but due to high computational costs of training an ensemble and finite available resources I had to compromise to five repetitions. The only exceptions here are the *Naive* and *Greedy* ensembles, for which I did not conducted repeated measurements. Both had to be constructed from a sufficiently large pool - in our case 25 agents - and in order to repeat measurements an even larger pool from which 25 agents could be sub-sampled would have had to be constructed. Again, due to the finite resources available, this was not a feasible option.

Table 1: Agent Task Completion on *map2seq*

	Validation	Test
Single	49.06 (48.10, 50.02)	44.21 (43.21, 45.21)
*Baseline**	52.90 (52.17, 53.62)	50.04 (48.12, 51.96)
Naive	54.82	50.86
$Greedy_{25}$	**56.12**	**52.57**
$Greedy_{6}$	53.24 (51.87, 54.61)	50.26 (49.29, 51.23)
$GNCL_{0}$	53.38 (53.03, 53.73)	51.96 (50.86, 53.06)
$GNCL_{0.05}$	53.14 (51.64, 54.64)	50.94 (49.53, 52.35)
$GNCL_{0.1}$	53.48 (51.81, 51.68)	50.30 (48.17, 52.43)
$GNCL_{0.5}$	50.72 (49.76, 51.68)	46.50 (45.53, 47.47)
$GNCL_{1}$	49.28 (48.29, 50.27)	43.06 (41.76, 44.36)

Note: Where repeated measurements have been taken, the 90% confidence interval is shown in parenthesis. Best result are marked in boldface. Confidence bounds of the *Single* agent are based on the 25 measurements available.
***** The ***Baseline*** measurement is based on only 5 random sampled ensembles (even though 200 are available) to keep ensemble reliability comparable. The estimation of the *Baseline* TC based on the 200 measurements is slightly better with a **validation TC of 53.04 (52.89, 53.20)** and a **test TC of 50.62 (50.46, 50.78)**.

Table 1 shows the task completion of the various ensembles. The $Greedy_{25}$ performs best, with the $GNCL_{0}$ trailing half a percentage point behind. Although the greedy search only selected three out of the 25 models, the ensemble still arguably utilized all 25 available models indirectly, putting it at an unfair advantage over the *GNCL* ensembles which are based on 6 models only. With both ensembles performing similarly good, we can formulate arguments in favour of either. The *Greedy* ensemble technique has lower

GPU requirements as one can train all individual models sequentially on the GPU. In the *GNCL* approach we need to train the models jointly and require therefore larger GPUs. If a large enough GPU is available, however, the *GNCL* becomes favourable as it achieves similar performance with much fewer trained models required. This argument can be further supported by the comparably weaker performance of the $Greedy_6$ ensemble, which is based on a subset of 6 trained models and is therefore a more appropriate comparison to the *GNCL* ensembles. If one is facing GPU size limitations for *GNCL* training one could furthermore quite easily implement the training algorithm to allow for distributed training across multiple GPU nodes.

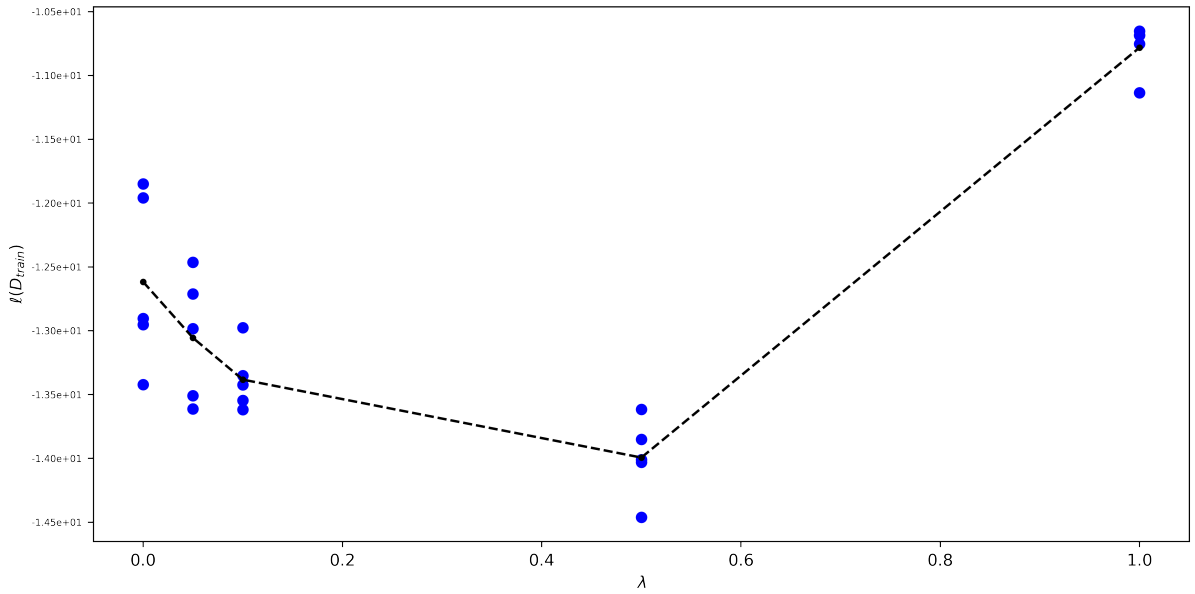
Among the GNCL ensembles we can observe worsening results with increasing regularizer λ. This confirms the observation made by Buschjäger et al. (2020) that enforced diversity can be hurtful for ensembles built out of high capacity neural networks. In the experiments conducted by Buschjäger et al. (2020), however, the evaluation metric remains relatively stable with increasing λ and only starts dropping off slowly at $\lambda > 0.7$. Buschjäger et al. (2020) argue that this is because 'higher capacity base models can achieve smaller losses on their own' and 'enforcing diversity reduces their performance and thus hurts their bias'. The results and discussion by Buschjäger et al. (2020) give the impression that GNCL generally increases ensemble performance, albeit with diminishing returns as the base learner capacity increases. Only with high λ it could become harmful at all. As table 1 shows, however, a harmful effect is visible right from the start with $\lambda > 0$ in our experiments. The ensembles even become significantly worse than the baseline with λ approaching one. The disappointing performance of *GNCL* ensembles raises the question if the conclusions drawn by Buschjäger et al. (2020) need to be reviewed.

A look at the extreme case of $\lambda = 1$ hints at an alternative interpretation of *GNCL*. In such case, the ensemble is trained in an end to end fashion. We technically no longer look at an ensemble of M models but instead at an model with M times the parameters with a somewhat restricted net architecture (M isolated networks only connected by a final averaging layer). When Buschjäger et al. (2020) demonstrate that a higher λ is favourable for ensembles of low capacity base models, they do not **necessarily** show the positive effect of model diversity through this. Instead, it is much more a demonstration that at this point it is more favourable to increase the model's complexity rather than ensembling it. Then, when we increase model complexity more and more, the returns diminish and eventually it becomes more favourable to ensemble models instead.

In our case it is clearly favourable to ensemble multiple models rather than increasing their parameters. According to this interpretation, however, the *GNCL* with $\lambda = 0$ should be indistinguishable from the baseline and not outperform it. The explanation to why this is not the case is actually just a small technicality during model selection. In the baseline case, we trained 6 models for 150 epochs, selected the checkpoint with best performance on the validation data of each model independently and then ensembled them. Whether those specific checkpoints work well together as an ensemble comes down just to pure chance. In the $GNCL_0$ case, however, we measured the performance of the ensemble as a whole at each checkpoint. Since the training loss has most often converged after 100 epochs we are left with 50 more checkpoints representing various combinations of 6 fully independently trained models. Instead of training the models jointly, we could also just store the last checkpoints of the models used for the baseline ensemble and then sample

50 combinations of which we keep the best one.

Among the GNCL ensembles, we can observe a negative correlation between λ and training loss up to $\lambda = 0.5$ (see figure 4). This observation supports the proposed interpretation of Generalized Negative Correlation Learning. The higher we set *lambda*, the more we shift from ensembling M models to getting an M times larger single model. An GNCL ensemble with $\lambda = 0.5$ receives partial control - at 50% - over the parameters of all its base agents and can therefore update the entire ensemble's set of parameters to fit the training data. In other words, it can force the base agents to land predictions whose average perfectly fits the training data. For low capacity models this could be beneficial, for high capacity models this results in over-fitting. Only the $\lambda = 1$ ensemble seems to not fit into the picture, as it has a much higher training loss than the rest of the GNCL ensembles. An explanation would be that with no incentive to fit the individual base agents to the target function, the restrictive architecture of the ensemble hinders model training.Essentially, each base model h_i needs to fit to some unknown function f_i^* so that the average of the base models fit the target function f^*. However, there are infinitely many combinations of functions f_i^* that satisfy this condition, which makes parameter convergence highly volatile. While Buschjäger et al. (2020) as well record a monotone decreasing relationship between λ and the train loss, they do not observe a similar collapse of model convergence at $\lambda = 1$.



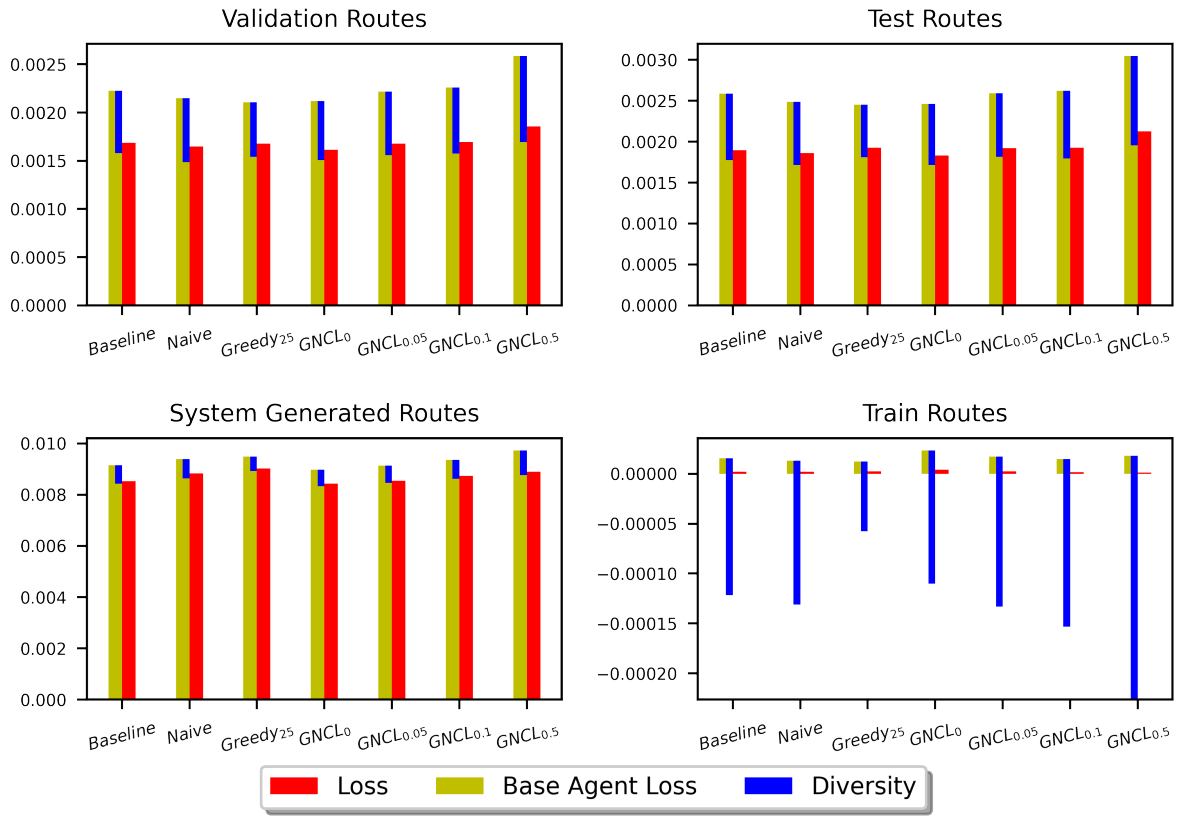
Note: Training loss $\ell(D_{\text{train}})$ is displayed in logarithmic scale for better visibility

Figure 4: *GNCL* Training Loss by λ

9.1.1 Loss Analysis

The implementation of equation 11 allows us to further analyse the loss of the various ensemble. Figure 5 display the ensembles' loss decomposition into the average loss of the base agents and the diversity term. Just as with the performances, the measurements are based on the average of five repeated trials. Quite noticeable is by how much

the loss approximation - the diversity term subtracted from the base agent loss - underestimates the actually measured Cross-Entropy loss of the ensemble prediction on the training data. This demonstrates, that even though the decomposition error is bounded, optimizing the objective 12 directly would be problematic. Essentially, we would have to take into account that the better the ensemble can fit the training data, the higher the decomposition error will become proportionally and the lower we would have to choose λ to compensate for this effect. Recall that the higher the capacity our base agents are, the better they can fit the training data, the lower we should choose λ. If we were to use objective 12 directly, we could no longer tell apart whether a certain low λ is favourable due to the base agents' high capacity or because it simply is necessary to compensate for the decomposition error.



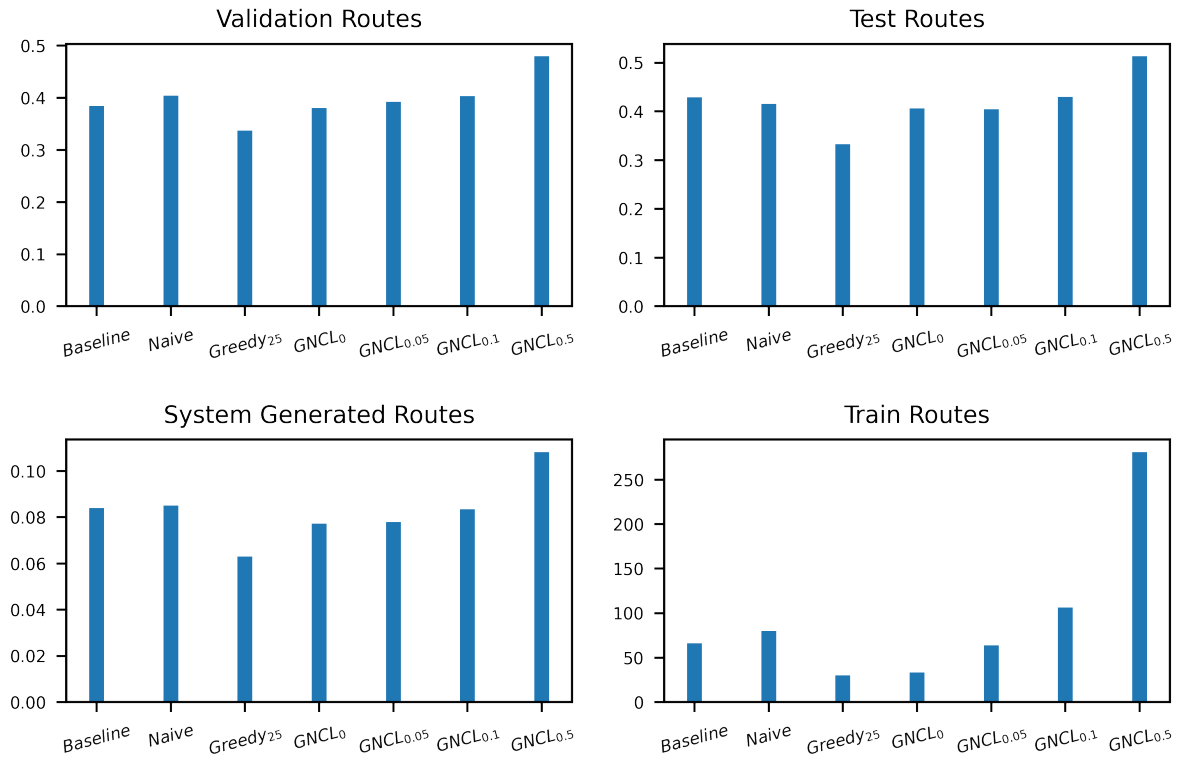
Note: The yellow bar displays the average base agent loss ℓ_{base}. The diversity term *Div* extends downwards from the average base agent loss so that the lower end equals the to total loss estimation according to the loss decomposition. The red bar displays the true measured loss. The difference between the lower end of the blue bar and the upper end of the red bar is therefore the decomposition error $\mathcal{R}$. $GNCL_1$ is excluded from the plots as it hinders readability due to its bad performance. See appendix 10 for full results.

Figure 5: Loss Decomposition by Dataset

Despite the substantial error, the decomposition of the train loss allows us to draw some conclusions. First and foremost, we can observe that the base agents of the *GNCL* ensembles indeed increase in prediction diversity with increased λ. This empirically verifies that the upper bound objective 14 indeed achieves the desired effect and allows us to enforce prediction diversity indirectly. While the diversities of the *GNCL* ensembles were to be expected, the three further ensembles provide interesting references. Surprisingly,

the best performing among the ensembles, $Greedy_{25}$, seems to have the lowest prediction diversity. However, we need to take into account if two ensembles have identical prediction diversity but achieve different total losses, the contribution of the diversity term to the total is different, relatively speaking. With increasing total loss even the highest diversity will eventually become insignificant in what it contributes to the loss overall. In order to get a better picture, we need to measure diversity relative to either the approximated or empirical loss. Since the loss approximated with equation 11 shows considerable errors on the train routes, we will choose the empirical loss instead.

As figure 6 shows, the $Greedy_{25}$ and $GNCL_0$ ensembles together have the lowest diversity in proportion to the ensemble loss and do so by a considerable margin. This is particularly interesting considering that in both cases, the base models were trained independently and then later on the combination of models performing best on the validation set were selected; either through checkpoint selection or greedy ensemble composition respectively. Unlike the other *GNCL* ensembles, the two in question are not trained with enforced diversity but can freely optimize the base agents and then afterwards simply select a combination of such fully optimized agents (or agent checkpoints) based on their diversity. Contrary to that, however, both ensembles nonetheless opt for less diverse base agents.



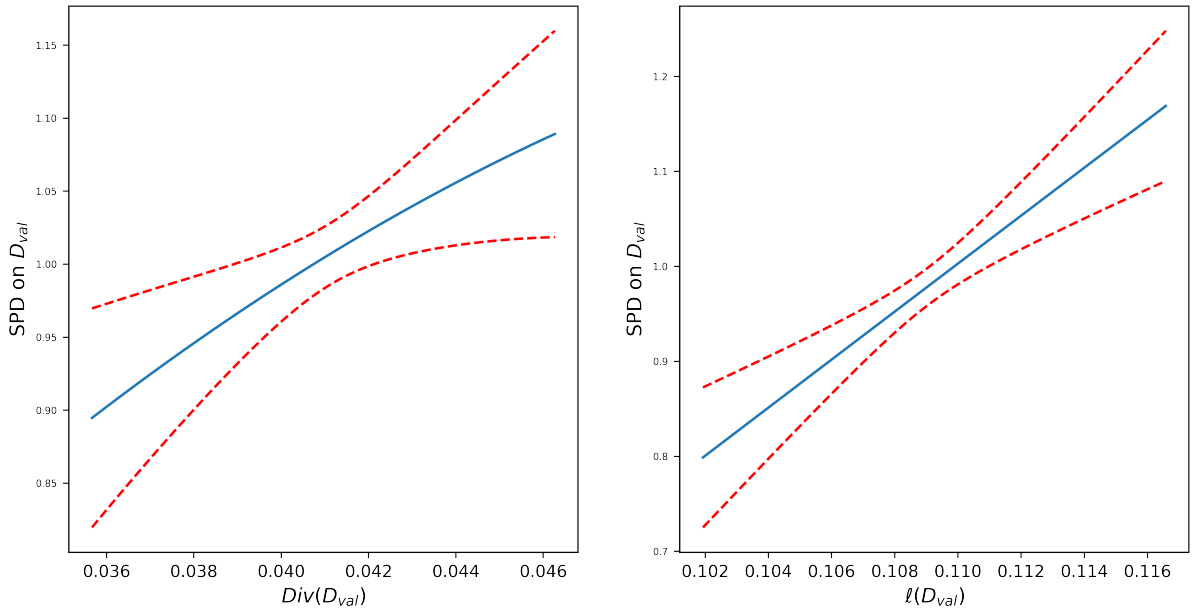
Note: The y-axis displays relative diversity: $\frac{Div}{TotalLoss}$. $GNCL_1$ is excluded from the plots as it hinders readability due to its bad performance. See appendix 11 for full results.

Figure 6: Diversity in Proportion to Total Loss

One explanation of why the $Greedy_{25}$ and $GNCL_0$ ensembles opt for lower diversity could be that the best agents are simply not that diverse. In order to increase diversity the ensembles would then have to include agents with worse validation set performance, which could have the costs outweigh the benefits. What speaks against this hypothesis is that only two out of three agents of the $Greedy_{25}$ ensemble are among the top six models with

best validation performance used in the *Naive* ensemble. We therefore know as a matter of fact that the $Greedy_{25}$ ensemble purposefully chose to include an agent with comparably weaker performance. Furthermore, we can observe that the *Naive* ensemble is much more diverse than the two ensembles, which invalidates the proposed explanation.

Another theory would be that since the two ensembles in question can select the desired combination of base agents out of of a large array of options, they could hypothetically actively fit the validation set. Given sufficiently many different predictors, one could purposefully select a subset whose average prediction minimizes the performance on the validation set. The diversity within the subset would not necessarily be a relevant criteria for the selection process. Consequently, one would over-fit the validation data and suffer increased generalization error on the test set. While the $Greedy_{25}$ ensemble experiences the largest loss increase (14.74%) between validation and test data, it does so only by an insignificant margin ($GNCL_{0.05}$ shows a 14.44% increase). Furthermore, table 1 shows an comparably average drop in task completion for $Greedy_{25}$ and the smallest performance drop among all models for $GNCL_0$. The proposed explanation can therefore not be supported through empirical evidence and needs to be discarded.



Spline	p-value	Sig. Code
$Div(D_{\text{val}})$	0.006	**
$\ell(D_{\text{val}})$	1.43e−5	**
intercept	1.11e−16	***
Pseudo R-Squared:	0.12	

Significance codes: ***: $p < 0.001$, **: $p < 0.01$:, *: $p < 0.05$

Figure 7: Effect of Diversity on Validation SPD-Metric at Constant Validation Loss

One remaining explanation to why the diversities of the $Greedy_{25}$ and $GNCL_0$ ensembles are low is that the base agents are chosen to optimize the SPD and not the overall loss (recorded through teacher forcing). This can make a decisive difference, as it is more favourable for the ensemble to be 100% accurate on 50% of the routes rather than being

even 90% accurate on 100% of the routes. Throughout a route, the agent needs to navigate on average 40 nodes. If the agent takes on each route on average even just one wrong turn within the first couple nodes, it will likely not be able to backtrack this error and wander far off the target node, which in turn will cause the SPD to increase substantially (note that low SPD is good). If we now have an ensemble of M agents of which each agent m would successfully complete a subset of validation routes $D_m \subset D_{val}$ then in theory it could occur that the ensemble as a whole only succeeds on the intersection $D_1 \cap D_2 \cap ... \cap D_M$. While the example provided here is obviously an extreme case, it is not unreasonable that high diversity could harm SPD in such way. Consequently, the ensemble should instead look for agents with high agreement rather than disagreement, so that it at least performs reliable on some subset of the routes.

The 200 random selected ensembles used for the analysis in section 5 show only a weak linear correlation score of Pearson's $r = -0.20$ between validation loss and task completion, which suggests that the $Greedy_{25}$ and $GNCL_0$ would not necessarily select a combination of base agents which achieves the best validation loss (through high diversity). The GAM smoothing spline in figure 7 even shows that at constant validation loss increased diversity indeed harms SPD, which would further support the last proposed explanation. However, the GAM in question measures a Pseudo R-Squared of only 0.12 and therefore attributes very little explanatory power to the two variables.

9.2 System Generated Routes

We can expect the ensembles' scores on the system generated routes (generated by the *graph-to-text* model) to differ substantially from their performance on the *map2seq* test data. While test and validation data are drawn from the same data distribution as the training data the ensembles were fit on, we can not state the same about the system generated routes by the *graph-to-text* model. Therefore, we can not expect the ensemble performing best on validation data to necessarily show the least generalization error on the system generated routes. From a theoretical perspective we are now measuring not the out-of-sample error, but the out-of-distribution error of the models.

A look at the results in 2 shows that the *Baseline* and *Naive* ensembles perform best. One could argue that this is due to the fact that both method do not contain a selection procedure that chooses the optimal checkpoint or agent combination based on the validation data performance. Since the other ensemble methods can optimize validation performance, they might attempt to over-fit the validation data in a way so that they achieve low SPD and therefore generalize worse to the out of distribution data.

The *Baseline* and *Naive* ensembles both show higher base agent diversity on the train routes than the $Greedy_{25}$ and $GNCL_0$ do. Additionally, among the *GNCL* ensembles we can see that some enforced diversity seems to benefit the ensemble. The best performance can be observed for the $\lambda = 0.05$ ensemble. Compared to the task completion on the test routes (see table 1), the performance drop off for higher λ's is also much less severe. This gives raise to the hope that we might finally observe some measurable benefit of enforcing ensemble diversity.

A closer investigation with help of the 200 random sampled ensembles shows no apparent benefit of training diversity (see figure 8). The training loss is a statistically significant

Table 2: Task Completion and Annotator-Success Prediction Accuracy on System Generated Routes

	TC	Annotator-Success Prediction Accuracy
Single	39.62 (39.12, 40.12)	63.92 (63.23, 64.61)
*Baseline**	**44.20 (43.10,45.30)**	66.94 (66.24, 67.64)
$Naive_{25}$	**44.14**	66.71
$Greedy_{25}$	43.86	66.57
$Greedy_{6}$	**44.05 (43.48, 44.61)**	**67.42 (66.45,68.39)**
$GNCL_{0}$	43.88 (43.27, 44.48)	66.51 (65.63, 67.39)
$GNCL_{0.05}$	**44.04 (43.33, 44.75)**	**67.31 (66.39, 68.23)**
$GNCL_{0.1}$	43.80 (42.84, 44.76)	66.77 (65.51, 68.04)
$GNCL_{0.5}$	42.80 (42.07, 43.53)	65.54 (65.37, 65.72)
$GNCL_{1}$	37.92 (35.73, 40.11)	64.31 (63.18, 65.45)

Note: Where repeated measurements have been taken, the 90% confidence interval is shown in parenthesis. Best result are marked in boldface. Confidence bounds estimation of *Single* agents is based on the 25 measurements available.
***** The ***Baseline*** measurement is based on only 5 random sampled ensembles (even though 200 are available) to keep ensemble reliability comparable. The estimation of the *Baseline* based on the 200 measurements shows no change in the **TC of 44.20 (44.12, 44.29)** and a slightly worse **accuracy of 66.91 (66.83, 67.00)**.

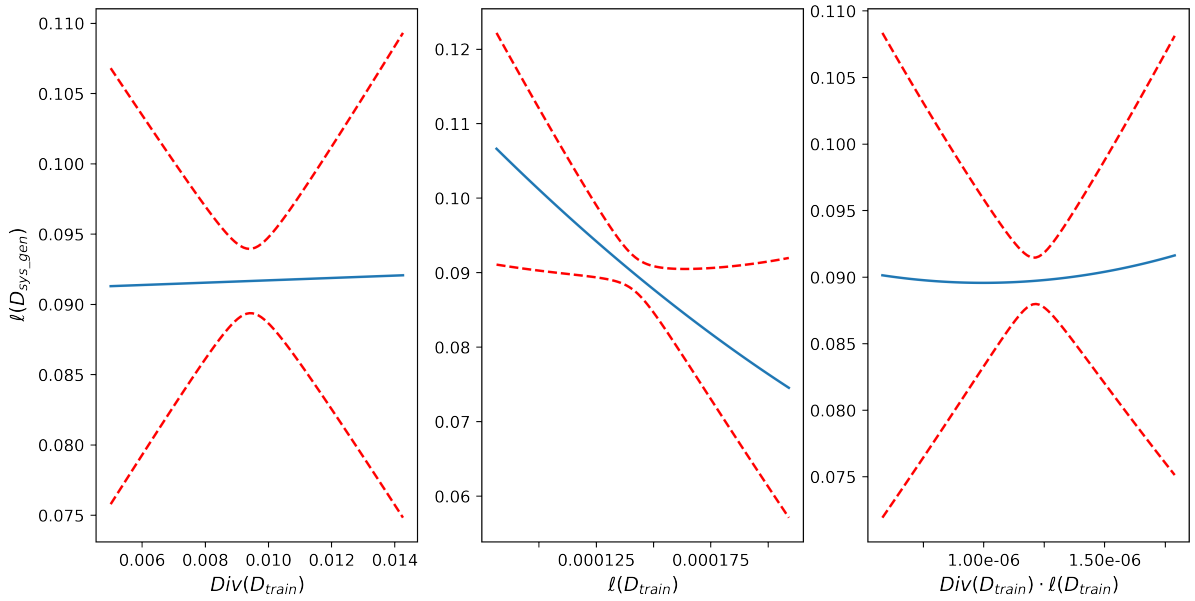
factor contributing to the loss on the system generated routes. The less (over-) fit the ensemble, the better it generalizes to the system generated routes. This insight, however, conflicts with the observation that the higher we choose λ, the better the ensemble fits the training data (see figure 4). Since the $GNCL_{0.05}$ has better training fit than the $GNCL_0$ ensemble, it should in turn achieve worse performance on the system generated routes, according to the GAM results. Since the confidence bounds for the splines in the GAM are too wide, we should not draw conclusions from it.

Since there is severe overlap between the confidence intervals of the task completions in table 2 and we cannot find other compelling empirical evidence, we should refrain from assuming a relation between training diversity and task completion on the system generated routes.

9.3 Agent-Annotator Correlation

We are not necessarily interested in the agent ensemble with best task completion on the system generated routes. If we wanted to achieve this, we would simply train our agent on such routes. Instead, we explicitly train the agent on realistic route descriptions compiled by human annotators so that - in an ideal case - the agent would succeed only on human-comprehensible instructions and fail otherwise.

While it does not make much sense intuitively, the task completion on system generated routes nonetheless serves as a better proxy for correlation with human annotators than the task completion on the human annotated test routes of the *map2seq* data does. Using the ensembles listed in table 2 as basis, test set task completion correlates with annotator agreement with Pearson's $r = 0.50$ while system generated route task completion achieves



Spline	p-value	Sig. Code
$Div(D_{\text{train}})$	1	
$\ell(D_{\text{train}})$	0.0179	*
$Div(D_{\text{train}}) \cdot \ell(D_{\text{train}})$	0.00306	**
intercept	1.11e−16	***
Pseudo R-Squared:	0.37	

Significance codes: ***: $p < 0.001$, **: $p < 0.01$:, *: $p < 0.05$

Figure 8: Effect of Diversity on Loss measured on System Generated Data

a correlation of $r = 0.67$. However, we should not conclude that we should therefore fit models to system generated directly rather than human compiled routes. By doing so, we would - other than by random chance - never exceed the majority baseline of 55.14% annotator agreement. However, if we train an ensemble on human annotated routes, instead, we can assume that improvements in task completion must predominantly stem from routes humans also succeed on.

The left plot on figure 9 displays the 200 baseline ensembles' recall of the 'annotator success' and 'annotator failure' classes against the ensembles' task completion. Naturally, the higher the task completion of the ensemble, the higher the recall for 'annotator success' and the lower the one for 'annotator failure'. While the increase in 'annotator success' recall is indeed much steeper than the decrease for the negative class, we nonetheless need to note down that there is some trade-off occurring.

To closer analyse this trade-off we can plot precision against task completion. we can see that the precision for the 'annotator success' class remains relatively stable at 75%, invariant of the task completion. If we assume this precision to stay fixed and then gradually - hypothetically - increase task completion rate, we would reach the maximal accuracy of 81.57% at a task completion of 73.57%. After this, the 'annotator success' class precision of 0.75% could not remain fixed as there would be no false negatives left to correctly classify as successes instead.Past this point, accuracy would drop again towards

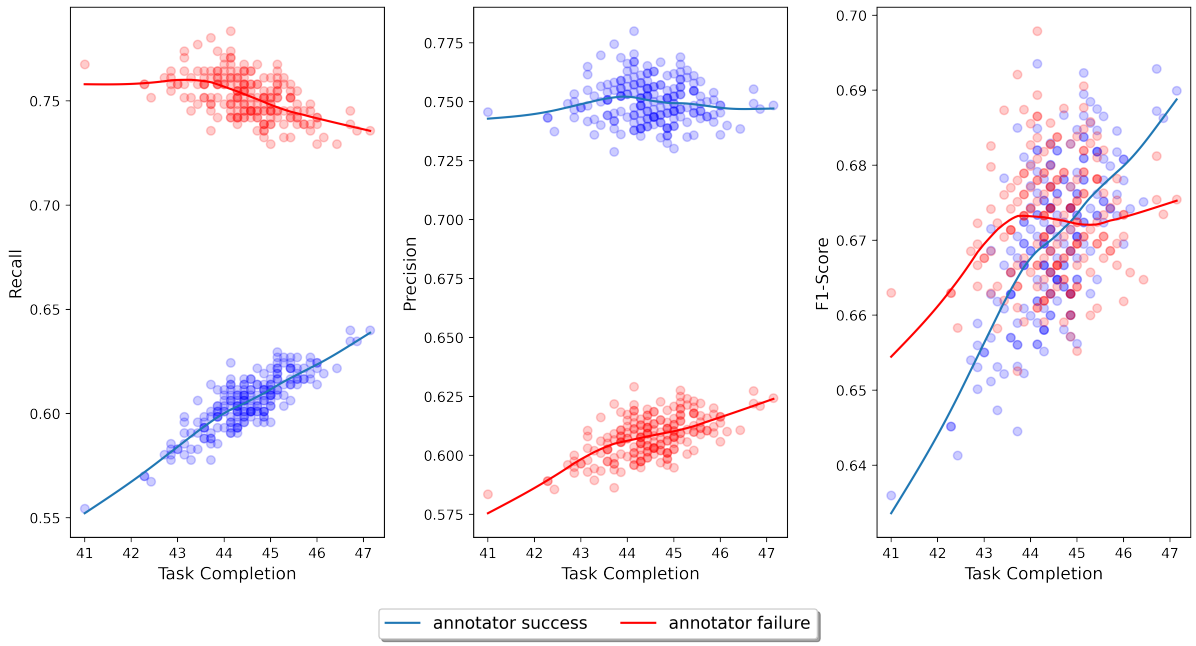


Figure 9: Recall, Precision and F1-Score by Task Completion

the majority baseline of 55%. Naturally, the fewer false negatives remain, the less likely would the model be able to maintain its 75% positive class precision. The extrapolation made here is therefore likely too optimistic and the real maximally achievable accuracy should be lower.

This upper bound for the accuracy is calculated based on the 200 baseline ensembles. However, all repeated measurements of each ensemble listed in table 2 (even of the $GNCL_1$) achieve around 75 ± 2 percent precision on the 'annotator success' class, invariant of task completion. This suggests that there is some sort of bias in the training data that does not allow even the best model to exceed this limit.

The precision for the 'success' class shows that one out of four agent successes cannot be replicated by human annotators. In order to succeed on routes the human annotators fail, the models must learn some information in the training data that the human annotators are not aware of or do not have access to. This information could be something as simple as the average number of nodes a route traverses on the navigation graph. The routes of the *map2seq* data-set traverse on average 40.02 (stdev. 3.19) nodes, the system generated routes quite similarly average at 40.26 (stdev. 3.23) nodes. Even if the system generated navigation instructions are incomprehensible, the agent will at least know, that it should traverse around 40 nodes, not much more or less. This knowledge puts the agent to an unfair advantage over the human annotators, especially if the *graph-to-text* model hallucinates instructions that would seem to lead beyond the navigation goal. One way of investigating this effect is to remove the step counter of the agent, the time step embedding t. If the $GNCL_{0.05}$ ensemble is trained without this information, average task completion drops from 44.04 down to 41.53 while annotator success prediction drops only from 67.31 to 66.40 accuracy. Although we cannot state this with statistical significance [8],

[8] I fit a linear regression with task completion as independent and accuracy as dependent variable over the 200 baseline ensembles and the ensembles in table 2. The original data points (a) and the results of the ensembles without step embedding (b) were then centered around the regressions' prediction

the modified agent ensemble achieves better annotator success prediction accuracy than ensembles with comparable task completion, suggesting that without the information, the ensembles predict more human-like.

Other than certain characteristic features of the routes, there could also be bias resulting from the *map2seq* navigation instructions. Since both the agent and the *graph-to-text* model are trained on the same data, they learn the same set of features. The *graph-to-text* learns to generate navigation instruction akin to the one in the *map2seq* data, while the agent learns to navigate based on the *map2seq* instructions. Since the *map2seq* instructions are created by human annotators, this should, in theory, not cause any issues. After all, if the *graph-to-text* learns to instruct like humans and the agent learns to navigate like humans, there should be no routes the agent can complete, but a human cannot. However, with only around six thousand training routes provided to the models, over-fitting can occur. As a consequence, the *graph-to-text* might generalize wrongly and, for instance, hallucinate instructions from similar training routes into the text. The agent, over-fit on the same data, could then nonetheless navigate the route successfully as it 'recalls' the same training routes and therefore knows what the *graph-to-text* model intended to describe.

Since the navigation instructions in the *map2seq* dataset are compiled based on an annotated map rather than directly on panorama images, the model also has an additional competitive advantage over human annotators. Imagine a T-junction with a hotel placed at the corner. When presented with the respective OpenStreetMap segment, the human annotators formulating navigation instructions would naturally use the hotel as a landmark and instruct the navigator to e.g. "turn left at the Plaza-Hotel". On the respective panorama image, however, the hotel might be hard to spot or not visible at all. Although the *map2seq* navigation instructions are validated, the validation criteria by Schumann and Riezler (2020) were rather loose. Each navigation instruction had to be completed by two annotators in the Street View environment, where it was sufficient that one of the two successfully completed the route to consider it as valid. Furthermore, the annotators were able to backtrack, which could allow them to find the right T-junction simply by process of elimination. Either way, a trained agent would not have any issues with the visibility, as it would simply over-fit the data sufficiently to know that this particular panorama image is 'the one with the hotel', regardless of whether it is visible or not. To account for this, we would have to investigate the agent-annotator correlation in the 'unseen' VLN setting, where the agent does not know the environment of the test-, or in this case system-generated, routes. However, the performance of outdoor VLN-actors plummets dramatically in an 'unseen' setting (Schumann and Riezler, 2022), which would not allow us to treat the agent ensembles as realistic beam-rerankers for the instruction generating graph-to-text model.

line. With help of a t-test, I checked for significance of the differences between the mean of group (a) and (b). While the mean of group (b) is indeed higher, the small sample size of 5 ensembles without step count embedding did not allow statistically significant conclusions.

10 Conclusion

One of the guiding principles when building an ensemble is to increase its diversity to improve generalization. However, diversity is not equally beneficial in every scenario or for every type of model. I have demonstrated that in a Vision-Language-Navigation setting, increased diversity shows no benefit for the agent-ensemble. In particular, I could reach following conclusions:

1. Diversity on the train data only then benefits loss on unseen data if the model is not already well fit to the training data. The better the model is fit, the less it benefits from diversity.

2. When diversity is enforced through a $GNCL$ learning algorithm, the ensemble increases training fit but loses generalization power due to overfitting. This effect can be observed for $\lambda < 0.5$ where at $\lambda = 0$ the base models are trained independently and at $\lambda = 1$ the ensemble is trained in an end to end fashion. For $\lambda = 1$, however, I recorded high training loss. With more GPU resources available, one could map out the range of λ to investigate whether there is a an area at $\lambda > 0.5$ with high diversity but less overfitting.

3. In a VLN task is it more valuable for an ensemble to have similar rather than diverse agents. The best performing ensembles purposefully selected models that have low prediction diversity. Much rather does the ensemble want to over-perform on a subset of routes with high confidence (through agreement of its base models) than it wants to perform only somewhat well on the entire set of routes. The latter case will inevitably result in worse VLN evaluation metrics than the former.

The experiments were conducted to not only investigate the effect of diversity, but also to test which type of VLN agent ensemble is optimal to predict human comprehension of artificial system generated navigation instructions and to investigate the overall feasibility of this idea. For this second task, I have come to following conclusions:

1. While the $GNCL_{0.05}$ performed well, it was only marginally better than the *baseline* ensemble method and worse than the $Greedy_6$. Empirical analysis on the baseline ensembles also showed no positive effect of base model prediction diversity.

2. Agent task completion on system generated data correlates with the agent-annotator success correlation. However, independent of the task completion, one out of four agent successes cannot be achieved by humans. Even if one could further increase the ensembles' task completion, the prediction accuracy would therefore never exceed 82%.

3. This upper bound of reachable accuracy can be explained through data bias. As the ensembles are trained on the same data as the *graph-to-text* model, it can memorize features that are incomprehensible or invisible to human annotators. Therefore, it can succeed on routes where humans fail.

Despite those observations, the ensembles substantially outperformed the majority baseline when predicting human annotator success on synthetic routes. A follow up paper should investigate by how much we can boost the performance of the *graph-to-text* model if we use an agent ensemble to re-rank the decoding beams by their likelihood that they can be completed by human annotators. While I cannot suggest a preferred ensembling method

for this task, I can at least state that ensemble diversity should not be taken into account when making this choice.

References

Brown, G. and L. I. Kuncheva (2010). "good" and "bad" diversity in majority vote ensembles. In N. El Gayar, J. Kittler, and F. Roli (Eds.), *Multiple Classifier Systems*, Berlin, Heidelberg, pp. 124–133. Springer Berlin Heidelberg.

Brown, G., J. Wyatt, and P. Tino (2005, 12). Managing diversity in regression ensembles. *Journal of Machine Learning Research 6*, 1621–1650.

Buschjäger, S., L. Pfahler, and K. Morik (2020). Generalized negative correlation learning for deep ensembling. *CoRR abs/2011.02952*.

Chen, H., A. Suhr, D. Misra, N. Snavely, and Y. Artzi (2018a). Touchdown: Natural language navigation and spatial reasoning in visual street environments.

Chen, H., A. Suhr, D. K. Misra, N. Snavely, and Y. Artzi (2018b). Touchdown: Natural language navigation and spatial reasoning in visual street environments. *CoRR abs/1811.12354*.

Cohen, J., P. Cohen, S. West, and L. Aiken (2002). *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences* (3rd ed.). Routledge. `https://doi.org/10.4324/9780203774441`.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Graves, A., S. Fernández, and J. Schmidhuber (2005, 01). Bidirectional lstm networks for improved phoneme classification and recognition. pp. 799–804.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S*, pp. 249–307. Routledge.

Krantz, J., E. Wijmans, A. Majumdar, D. Batra, and S. Lee (2020). Beyond the navgraph: Vision-and-language navigation in continuous environments.

Kuncheva, L. and C. Whitaker (2003, 05). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning 51*, 181–207.

Liu, Y. and X. Yao (1999). Ensemble learning via negative correlation. *Neural Networks 12*(10), 1399–1404.

Melville, P. and R. Mooney (2005, 03). Creating diversity in ensembles using artificial data. *Information Fusion 6*, 99–111.

Opitz, M., H. Possegger, and H. Bischof (2017, 03). Efficient model averaging for deep neural networks. pp. 205–220.

Perrone, M. and L. Cooper (1993, 08). When networks disagree: Ensemble methods for hybrid neural networks. *Neural networks for speech and image processing*.

Perrone, M., T. Michael, P. Perrone, P. Leon, L. Cooper, N. Intrator, and P. Elbaum (1994, 03). Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2014). Imagenet large scale visual recognition challenge.

Schumann, R. and S. Riezler (2020). Generating landmark navigation instructions from maps as a graph-to-text problem. *CoRR abs/2012.15329*.

Schumann, R. and S. Riezler (2022). Analyzing generalization of vision and language navigation to unseen outdoor areas.

Servén, D. and C. Brummitt (2018, March). pygam: Generalized additive models in python.

Tom, A. and M. Denis (2004, 12). Language and spatial cognition: Comparing the roles of landmarks and street names in route instructions. *Applied Cognitive Psychology 18*, 1213 – 1230.

Webb, A., C. Reynolds, D.-A. Iliescu, H. Reeve, M. Luján, and G. Brown (2019, 02). Joint training of neural network ensembles.

11 Appendix

11.1 Derivations

11.1.1 Cross Entropy Loss - Decomposition Error

Let $z \in \mathbb{R}^C$ be the model output containing the class prediction scores z_1 to z_C and $y \in \mathbb{R}^C$ be the class target scores. Through the softmax function we map each score z_i to a probability:

$$q_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

The Cross Entropy Loss ℓ is then calculated as follows:

$$\ell(z, y) = -\sum_{i=1}^{C} y_i \log(q_i) = -\sum_{i=1}^{C} y_i \log\left(\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\right)$$

The entries in the Gradient are:

$$\frac{\partial \ell}{\partial z_i} = q_i - \mathbb{1}\{y_i = 1\}$$

The entries of the Hessian are:

$$\frac{\partial^2 \ell}{\partial z_i, \partial z_j} = q_i(\mathbb{1}\{i = j\} - q_j)$$

Finally, the entries of the third derivative tensor can be calculated as:

$$\begin{aligned}\frac{\partial^3 \ell}{\partial z_i, \partial z_j, \partial z_k} =& \mathbb{1}\{i = j\} q_i(\mathbb{1}\{i = k\} - q_k) \\ &- q_i q_j(\mathbb{1}\{i = k\} - q_k) \\ &- q_i q_j(\mathbb{1}\{j = k\} - q_k)\end{aligned}$$

According to the softmax function, we know that $\sum_{c=1}^{C} q_c = 1$ and $0 \leq q_c \leq 1$. Buschjäger et al. (2020) conclude that therefore the maximum entry in the third derivative tensor occurs for pairwise unequal i, j, k with $q_i = q_j = q_k = \frac{1}{3}$. To clarify, I will demonstrate all possible cases and show how this conclusion is reached:

Case $i = j = k$:

$$\begin{aligned}\frac{\partial^3 \ell}{\partial z_i^3} &= q_i(1 - q_i) - q_i^2(1 - q_i) - q_i^2(1 - q_i) \\ &= q_i(1 - q_i) - 2q_i^2(1 - q_i) \\ &= 2q_i^3 - 3q_i^2 + q_i \leq 0.0962\end{aligned}$$

Case $i = j \neq k$ (since the order in which the derivatives are taken is irrelevant, this is equivalent to $i \neq j = k$ and $i = k \neq j$):

$$\begin{aligned}
\frac{\partial^3 \ell}{\partial z_i^2, \partial z_k} &= q_i(0 - q_k) - q_i^2(0 - q_k) - q_i^2(0 - q_k) \\
&= -q_k(q_i - 2q_i^2) \qquad \text{substitute } q_k = 1 - 2q_i \\
&= (2q_i - 1)(q_i - 2q_i^2) \\
&= -4q_i^3 + 4q_i^2 - q_i \leq 0
\end{aligned}$$

Case $i \neq j \neq k$:

$$\begin{aligned}
\frac{\partial^3 \ell}{\partial z_i, \partial z_j, \partial z_k} &= -q_i q_j(0 - q_j) - q_i q_j(0 - q_j) \\
&= -2(q_i q_j(0 - q_k)) = 2q_i q_j q_k \leq 0.075
\end{aligned}$$

Maximizing the product $2q_iq_jq_k$ is equivalent to maximizing the sum $\log(q_i) + \log(q_j) + \log(q_k)$. With the constraint that $q_i + q_j + q_k = 1$ we know that this sum is maximal if $q_i = q_j = q_k = \frac{1}{3}$. Since the logarithm is concave, deducting some ϵ from one q to add it to another will always result in a smaller sum. At maximum, we therefore get $2 \cdot \frac{1}{3^3} \approx 0.075$. As the partial third derivatives have an upper limit, the overall decomposition error of 9 is therefore bounded for the Cross Entropy Loss.

11.2 Figures

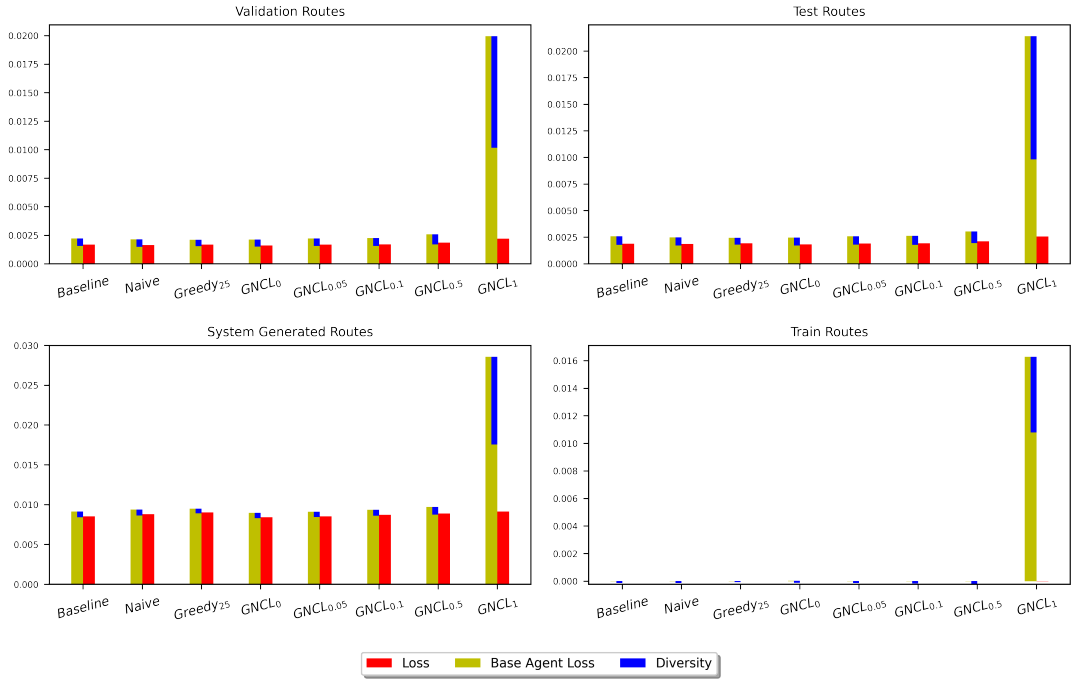


Figure 10: Loss Decomposition by Dataset

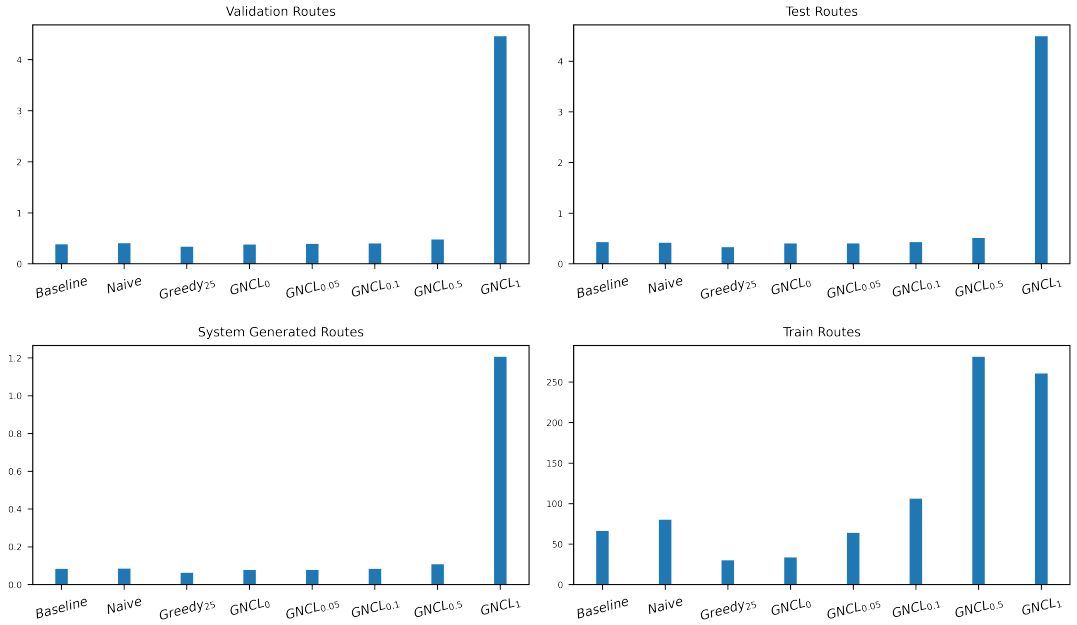


Figure 11: Diversity in Proportion to Total Loss

11.3 Tables

Agent ID	Validation SPD	Validation TC	Test TC	in *Naive*	in $Greedy_{25}$
19	4.6	50.5	47.9	x	x
9	4.7	53.5	50.1	x	x
18	5.2	49.4	46.1	x	
24	5.2	51.0	47.0	x	
15	5.3	51.1	43.4	x	
20	5.3	49.7	42.3	x	
2	5.4	50.2	43.1		
3	5.4	50.6	47.9		
7	5.4	50.2	43.1		
8	5.4	50.6	47.9		
12	5.4	52.3	45.1		
16	5.4	49.8	46.6		
10	5.5	48.9	42.7		
13	5.5	48.2	41.7		x
1	5.6	52.3	43.4		
6	5.6	52.3	43.4		
11	5.6	48.7	46.6		
22	5.6	49.0	45.0		
17	5.7	48.2	44.7		
4	5.8	45.2	38.3		
14	5.8	47.3	44.7		
21	5.8	46.9	41.4		
0	6.3	42.6	39.4		
5	6.3	42.6	39.4		
23	6.8	45.5	44.1		

Table 3: Base Agent Metrics