1. Notation

Notations used in multiple sections of this paper are defined here for fast reference. Auxiliary tasks $\phi_1, \phi_2$ and the main VLN task $\phi_{VLN}$ constitute the set of tasks $\Phi$. Inputs and embeddings are specified as $l$ (linguistic), $v$ (visual), and $\eta$ (multimodal). A complete textual instruction is denoted as $\tau$, $\varsigma$ is a span, and $\psi$ is a perspective. Linguistic and visual inputs for the PM-VLN are denoted as $(i_t', \psi_t)$ and embeddings processed in prioritisation operations are $(e_l, e_v)$. In contrast, $U$ denotes a set of embeddings from the main model, which are derived from inputs $(\bar{e}_\eta, \psi_{cat})$. The notations $\Delta$ and $\Theta$ are respectively visual boost filtering and self-attention operations. Table 1 provides a reference source for standard notation appearing throughout this paper. Other notations are defined in the sections where they are used.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Usage in this paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Matrix</td>
</tr>
<tr>
<td>$AA$</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>$B, b$</td>
<td>Bias</td>
</tr>
<tr>
<td>$D$</td>
<td>Dataset</td>
</tr>
<tr>
<td>$Train, Dev, Test$</td>
<td>Dataset partitions</td>
</tr>
<tr>
<td>$\exists$</td>
<td>Exists</td>
</tr>
<tr>
<td>$\forall$</td>
<td>For every (eg member in a set)</td>
</tr>
<tr>
<td>$g$</td>
<td>Function</td>
</tr>
<tr>
<td>$H$</td>
<td>Hypothesis</td>
</tr>
<tr>
<td>$L$</td>
<td>Layer of a model</td>
</tr>
<tr>
<td>$\text{len}$</td>
<td>Length</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Mean</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of samples</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Or</td>
</tr>
<tr>
<td>$P$</td>
<td>Probability</td>
</tr>
<tr>
<td>$q$</td>
<td>Algorithm</td>
</tr>
<tr>
<td>$S$</td>
<td>Signal detected</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Set of parameters</td>
</tr>
<tr>
<td>$W, w$</td>
<td>Set of weights</td>
</tr>
<tr>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$\triangleq$</td>
<td>Equal by definition</td>
</tr>
</tbody>
</table>

Table 1: Reference List for Standard Notation.
2. Additions to the Method Section

2.1. Theoretical Basis for Cross-modal Prioritisation

This section provides a theoretical basis for a hierarchical process of cross-modal prioritisation that optimises attention over linguistic and visual inputs. In this section we use $q$ to denote this process for convenience. During the main task $\phi_{VLN}$, $q$ aligns elements in temporal sequences $\tau$ and Route and localises spans and visual features w.r.t. a subset of all entities $\text{Ent}$ in the routes:

$$q = \| x_l - x_v \| \rightarrow \max_{\text{subject to } P_{D_{\text{Ent}}}[\tau, \text{Route}] \leq R}$$

\hspace{1cm} (1)

Inputs in $\phi_{VLN}$ consist of a linguistic sequence $\tau$ and a visual sequence Route for each trajectory $j$ in a set of trajectories. As a result of the process followed by Chen et al. [4] to create the Touchdown task, these inputs conform to the following definition.

Definition 1 (Sequences refer to corresponding entities). At each step in $j$, $|x_l|$ and $|x_v|$ are finite subsequences drawn from $\tau_j$ and $\text{Route}_j$ that refer to corresponding entities appearing in the trajectory $\text{ent}_j \in \text{Ent}$.

In order to simplify the notation, these subsequences are denoted in this section as $x_l$ and $x_v$. Touchdown differs from other outdoor navigation tasks [5] in excluding supervision on the alignment over cross-modal sequences. Furthermore $\text{len}(\tau_j) \neq \text{len}(\text{Route}_j)$ and there are varying counts of subsequences and entities in trajectories. In an approach to $\phi_{VLN}$ formulated as supervised classification, an agent's action at each step $\alpha_t \equiv \text{classification } c_t \in \{0, 1\}$ where $c$ is based on corresponding $\text{ent}_t$ in the pair $(x_l, x_v)_t$. The likelihood that $c_t$ is the correct action depends in turn on detecting $S$ signal in the form of $\text{ent}_t$ from noise in the inputs. The objective of $q$ then is to maximise $P_S$ for each point in the data space.

The process $q$ is composed of multiple operations to perform two functions of high-level alignment $g_{\text{Align}}$ and localisation $g_{\text{Loc}}$. At the current stage $\text{stg}$, function $g_{\text{Align}}$ selects one set of spans $\varphi_{\text{stg}} \in (\varphi_1, \varphi_2, \ldots, \varphi_n)$

\[
\begin{align*}
\text{stg} = \begin{cases} 
\text{Start}, & \text{if } t = 0 \\
\text{End}, & \text{if } t = -1 \\
\forall \text{stg}_{\text{other}}, & n \in N \sum_{n=1}^{t-1} > t-1 \text{ otherwise.}
\end{cases}
\end{align*}
\]

This is followed by the function $g_{\text{Loc}}$, which predicts one of $\text{scnt}_0 \lor \text{scnt}_{t-1}$ as the span $\varsigma$ relevant to the current trajectory step $\text{scnt}$

\[
\begin{align*}
\text{scnt} = \begin{cases} 
\text{scnt}_0, & (\tau, \psi_0) = 0 \\
\text{scnt}_{t-1}, & \text{otherwise.}
\end{cases}
\end{align*}
\]

We start by describing the learning process when the agent in $\phi_{VLN}$ is a transformer-based architecture $\text{Enc} + \text{Clas}$ excluding an equivalent to $q$ (e.g. VisualBERT in Table 1 of the main report). $\text{Enc} + \text{Clas}$ is composed of two core subprocesses: cross-modal attention to generate representations $q(\bigoplus(L \iff V))$ and a subsequent classification $\text{Clas}(c_n)$.

Definition 2 (Objective in $\text{Enc} + \text{Clas}$). The objective $\text{Obj}_j(\theta)$ for algorithm $q(\bigoplus(L \iff V))$, where $L$ and $V$ are each sequences of samples $\{x_1, x_2, \ldots, x_n\}$, is the correspondence between samples $x_l$ and $x_v$ presented at step $t$ in $\sum_{i=1}^{t} t_i = t_1 + t_2 + \ldots + t_n$.

It is observed that in the learning process for $\text{Enc} + \text{Clas}$, any subprocesses to align and localise finite sequences $x_l$ and $x_v$ w.r.t. $\text{ent}_j$ are performed as implicit elements in the process of optimising $\text{Obj}_j(\theta)$. In contrast the basis for the hierarchical learning process enabled by our framework $\text{FL}_{PM}$ - which incorporates $q_{PM}$ with explicit functions for these steps - is given in Theorem 1.

Theorem 1. Assuming $x_l$ and $x_v$ conform to Definition 1 and that $\forall x \in L \exists x \in V$, an onto function $g_{\text{Map}} = mx + b, m \neq 0$ exists such that:

$$g_{\text{Map}}(x_l, x_v) \rightarrow \max \left[ \text{ent}^{(x_l; x_v)}_j \in \text{Ent} \right]$$

\hspace{1cm} (2)

In this case, additional functions $g_{\text{Align}}$ and $g_{\text{Loc}}$ - when implemented in order - maximise $g_{\text{Map}}$:

$$\max \left[ \text{subject to } (g_{\text{Align}}, g_{\text{Loc}}, g_{\text{Map}}) \forall \text{ent}^{(x_l; x_v)}_j \in L_j \cap V_j \right]$$

\hspace{1cm} (3)

Remark 1 Let $P(\text{max } g_{\text{Map}})$ in Theorem 1 be the probability of optimising $g_{\text{Map}}$ such that the number of pairs $N^{(x_l; x_v)}_i$ corresponding to $\text{ent}_j \in L_j \cap V_j$ is maximised. It is noted that $N^{(x_l; x_v)}_i$ is determined by all possible outcomes in the set of cases $\{(x_l, x_v) \Rightarrow \text{ent}_j, (x_l, x_v) \Leftrightarrow \text{ent}_j, x_l \Leftrightarrow x_v\}$. As the sequences of instances $i$ in $x_l, x_v$ and $\text{ent}_j$ are forward-only, it is also noted that $N^{(x_l; x_v)}_{i+1} < N^{(x_l; x_v)}_i$ if $\text{ent}_i \notin x_l, \text{ent}_i \notin x_v$, or $\text{ent}_i \neq \text{ent}_v$. By definition, $N^{(x_l; x_v)}_{i+1} > N^{(x_l; x_v)}_i$.
if \( P(\text{ent}_i = x_i = x_{e_i}) \) - where the latter probability is s.t. processes performed within finite computational time \( CT(n) \) - which implies that \( P(\text{max} \ g_{\text{Map}} | P(\text{ent}_i = x_i = x_{e_i}) \).

**Remark 2.** Following on from Remark 1, \( CT(nP(\text{ent}_i=x_i=x_{e_i})) \) when \( q \) contains \( g_t \) and function \( g_t(\text{max}(N(x_i,x_i)) \Rightarrow \text{ent}_j \in L_j \cap V_j) \), where \( g_t \in G < CT(nP(\text{ent}_i=x_i=x_{e_i})) \) when \( q \) does not contain \( g_t < CT(nP(\text{ent}_i=x_i=x_{e_i})) \) when \( q \) contains \( g_t \) and function \( g_t(\text{max}(N(x_i,x_i)) \neq \text{ent}_j \in L_j \cap V_j) \).

**Discussion** In experiments, we expect from Remark 1 that results on \( \phi_{\text{VLN}} \) for architectures such as \( \text{Enc} + \text{Clas} \) - which exclude operations equivalent to those undertaken by the onto function \( g_{\text{Map}} \) - will be lower than the results for a framework \( \text{FLPM} \) over a finite number of epochs. We observe this in Table 1 of the main report when comparing the performance of respective standalone and + \( \text{FLPM} \) for VisualBERT and VLN Transformer systems. Poor results for variants (a) and (b) in Tables 2 and 3 of the main report in comparison to \( \text{FLPM} \) + VisualBERT(4) also support the expectation set by Remark 2 that performance will be highly impacted in an architecture where operations in \( g_{\text{Map}} \) increase the number of misalignments.

**Proof of Theorem 1** We use below \( q^* \) for a generic transformer-based system that predicts \( \alpha \) on \( (L,V) \), \( \nabla x \) for gradients, and \( \Theta^{a*} \) to denote \( \Theta^{\text{Enc}+\text{Clas}} \nu \Theta^{\text{Enc}+q} \). Let sequence \( x_i = [\text{ent}_1, \text{ent}_2, \ldots, \text{ent}_{n_1}] \) and sequence \( x_v = [\text{ent}_1, \text{ent}_2, \ldots, \text{ent}_{n_2}] \), where \( n_1 \) and \( n_2 \) are unknown. Furthermore at any point during learning, \( P_S(x_i,x_v) \) is spread unevenly over \( \text{ent}_j \) in relation to \( \Theta^{a*} \approx \chi \).

**Propositions** We start with the case that \( \exists \ ent_j : \text{ent}(x_i) \) and \( \text{ent}(x_v) \). \( CT(n_{\text{Ent} \in L \cap V}) \) for \( \Theta^{a*+g_t} < CT(n_{\text{Ent} \in L \cap V}) \) for \( \Theta^{a*} \) where \( g_t \) accounts for \( \Delta(\text{Ent}_1,\text{Ent}_2) \). We next consider the case where \( \nexists \ ent_j : \text{ent}(x_i) \nu \text{ent}(x_v) \). Where \( \nexists \text{ Loc} \) then \( P_S(x_i,x_v) \approx \text{ent}(x_v) \). We conclude with the case where \( \exists \text{ Ent} : x_i \nu x_v. \)

As \( (\text{Ent}_L,\text{Ent}_V) \Rightarrow \text{Ent}, \Theta^{a*} \approx \text{max}(N(x_i,x_v)) \in \chi \). \( P_S(x_i,x_v) \) where \( \text{ent}_i = x_i = x_{e_i} > \text{ent}_i \in \Theta^{a*} \approx \text{max}(N(x_i,x_v)) \). Furthermore \( P \exists \text{ ent} \in \approx (\text{ent}_i) > \nexists \text{ ent} \neq \text{ent}_i \). Therefore slope \( \nabla x \) increases and \( CT(n_{\text{Ent} \in L \cap V}) \) for \( \Theta^{a*+q} < CT(n_{\text{Ent} \in L \cap V}) \).

2.2. Visual Boost Filtering

We provide further description on the initial operations conducted during feature-level localisation. Parameterised visual boost filtering as proposed by Caranza et al. [3] is applied to perspectives. Let \( \text{ConvBF} \) be a convolutional layer with a kernel \( \kappa \) and weights \( W \) that receives as input \( \psi_i \). In the first operation \( g_{\text{USM}} \), a Laplacian of Gaussian kernel \( \kappa_{\text{LoG}} \) is applied to \( \psi_i \). The second operation \( g_{\text{BF}} \) consists of subtracting the output \( e_v \) from the original tensor \( \psi_i \):

\[
g_{\text{BF}}(e_v) = (\lambda - 1)(e_v) - g_{\text{USM}}(\psi_i)
\]  

(4)

where \( \lambda \) is a learned parameter for the degree of sharpening.

A combination of \( g_{\text{USM}} \) and \( g_{\text{BF}} \) is equivalent to adaptive sharpening of details in an image with a Laplacian residual [2]. Here operations are applied directly to \( e_v \) and adjusted at each update of the convolutional layer with a parameterised control \( \beta \lambda \). In the simple and efficient implementation from [3], \( \sigma \) in the distribution \( \text{LoG}(\mu_j,\sigma_j) \) is fixed and the level of boosting is reduced to a single learned term

\[
\Delta z(x_1,x_2) = \beta \lambda \left( \sum_j (AA'_{\kappa_{j}} - A_{W_{\kappa_{j}}})z \right)
\]

(5)

where \( A_W \) is a matrix of parameters and \( AA' \) is the identity.

3. Datasets

3.1. Generation and Partition Sizes

The MC-10 dataset consists of visual, textual and geospatial data for landmarks in 10 US cities. We generate the dataset with a modified version of the process outlined by [1]. Two base entity IDs - Q2221906 (“geographic location”) and Q83620 (“thoroughfare”) - form the basis of queries to extract entities at a distance of \( <= 2 \) hops in the Wikidata knowledge graph. Constituent cities consist of incorporated places exceeding 1 million people ranked by population density based on data for April 1, 2020 from the US Census Bureau. Images and coordinates are sourced from Wikimedia and text summaries are extracted with the MediaWiki API. Geographical cells are generated using the S2 Geometry Library [4] with a range of \( n \) entities [1, 5]. Statistics for MC-10 are presented by partition in Table 2. As noted above, only a portion of textual inputs are used in pretraining and experiments.

Table 2: Statistics for the MC-10 dataset by partition.

<table>
<thead>
<tr>
<th>Train</th>
<th>Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,100</td>
<td>955</td>
</tr>
<tr>
<td>727</td>
<td>745</td>
</tr>
</tbody>
</table>

TR-NY-PIT-central is a set of image files graphing path traces for trajectory plan estimation in two urban areas. Trajectories in central Manhattan are generated from routes in the Touchdown instructions. Links $E$ connecting $O$ in the Pittsburgh partition of StreetLearn are the basis for routes where at least one node is positioned in the bounding box delimited by the WGS84 coordinates $(40° 27’ 38.82”$, $-80° 1’ 47.85”) and $(40° 26’ 7.31”, -79° 59’ 12.86”)$. Labels are defined by step count $cnt$ in the route. Total trajectories sum to 9,325 in central Manhattan and 17,750 in Pittsburgh. In the latter location, routes are generated for all nodes with 50 samples randomly selected where $cnt = < 7$ and 200 samples where $cnt > 7$. The decision to generate a lower number of samples for shorter routes was determined by initial tests with the ConvNeXt Tiny model. We opt for a maximum $cnt$ of 66 steps to align with the longest route in the training partition of Touchdown. The resulting training partition of samples for Pittsburgh consists of 17,000 samples and is the resource used to pretrain $g_{PMTP}$ in the PM-VLN module.

3.2. Samples from Datasets

In auxiliary task $\phi_2$, the $g_{PMF}$ submodule of PM-VLN is trained on visual, textual, and geodetic position data types. Path traces from the TR-NY-PIT-central are used in $\phi_1$ to pretrain the $g_{PMTP}$ submodule on trajectory estimation. Samples for entities in MC-10 and path traces in TR-NY-PIT-central are presented in Figures 1 and 2.
4. Code and Data

Source code for the project and instructions to run the framework are released and maintained in a public GitHub repository under MIT license (https://github.com/JasonArmitage-res/PM-VLN). Code for the environment, navigation, and training adheres to the codebases released by [8] and [4] with the aim of enabling comparisons with benchmarks introduced in prior work on Touchdown. Full versions of the MC-10 and TR-NY-PIT-central datasets are published on Zenodo under Creative Commons public license (https://zenodo.org/record/6891965#.YtwoS3ZBxD8).

5. Acknowledgments

This publication is supported by the Digital Visual Studies program at the University of Zurich and funded by the Max Planck Society. RS has received funding by the Swiss National Science Foundation (project MUTAMUR; no. 176727). The authors would like to thank Howard Chen, Piotr Mirowski, and Wanrong Zhu for assistance and feedback on questions related to the Touchdown task, data, and framework evaluations.

References