Latent Group Dropout for Multilingual and Multidomain Machine Translation
Minh Quang Pham, Josep Crego, François Yvon

To cite this version:
Minh Quang Pham, Josep Crego, François Yvon. Latent Group Dropout for Multilingual and Multidomain Machine Translation. Annual Meeting of North American Chapter of the ACL (NAACL), Association for Computational Linguistics, Jul 2022, Seattle, United States. hal-03720395

HAL Id: hal-03720395
https://hal.archives-ouvertes.fr/hal-03720395
Submitted on 11 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Latent Group Dropout for Multilingual and Multidomain Machine Translation

MinhQuang Pham†‡*, Josep Crego†, François Yvon‡
†SYSTRAN / 5 rue Feydeau, 75002 Paris, France
firstname.lastname@systrangroup.com
‡Université Paris-Saclay, CNRS, LIMSI, 91405 Orsay, France
firstname.lastname@limsi.fr

Abstract
Multidomain and multilingual machine translation often rely on parameter sharing strategies, where large portions of the network are meant to capture the commonalities of the tasks at hand, while smaller parts are reserved to model the peculiarities of a language or a domain. In adapter-based approaches, these strategies are hardcoded in the network architecture, independent of the similarities between tasks. In this work, we propose a new method to better take advantage of these similarities, using a latent-variable model. We also develop new techniques to train this model end-to-end and report experimental results showing that the learned patterns are both meaningful and yield improved translation performance without any increase of the model size.

1 Introduction
Multidomain and multilingual machine translation aim to develop one single model to perform translation for multiple domains and multiple language pairs, respectively. These paradigms are motivated by the compactness of the resulting translation system (Chu and Dabre, 2018; Dabre et al., 2020), the hypothetical positive knowledge transfer between similar domains (Pham et al., 2021) or between languages in the same family (Tan et al., 2019). However, having all the tasks use exactly the same model parameters can cause negative interference between unrelated tasks (Conneau et al., 2020; Wang et al., 2020b). Hence, the recent development of approaches relying on a partial sharing of the parameters, e.g. using adapter layers as studied in (Houlsby et al., 2019; Bapna and Firat, 2019; Pham et al., 2020; Philip et al., 2020). If these techniques have proven effective for building strong baselines, they fail to fully take advantage of the similarities that exist between domains and tasks, as documented eg. in (Pham et al., 2021). This is because the partition of the parameter space between generic or task-specific subparts, and their allocation to each task, is hard-coded in the network, irrespective of the actual commonalities and differences in the data space.

In this work, we study and develop a new method, multi-task group dropout, aimed to take into account the similarity between tasks in a more effective way, by learning the network organization from the data. To this end, we introduce a set of latent variables in the model, to account for the unseen association between tasks and regions of the representation space and show how training can still be performed end-to-end using a variational surrogate of the log-likelihood loss function. Our experiments with multilingual and multidomain machine translation confirm that this method can automatically detect similarities in the data, meaning that related tasks use the same subparts of the network. Our results also show that this method is comparable to using adapter layers in a number of empirical comparisons; however, contrarily to adapters, these performance are obtained without any increase of the model size. Our contributions are primarily methodological and can be summarized as follows:

1. We introduce a novel, sound mathematical formulation to the problem of jointly learning task-dependent sub-networks and the parameters of the underlying models using variational probabilistic modeling techniques;
2. We present algorithms to train this model end-to-end with very little extra parameters;
3. We report, using an extensive set of experiments, gains for multidomain MT and very low-resourced languages in multilingual MT;
4. We study how this method can actually exploit the similarities between tasks to learn
Figure 1: Latent group dropout. The set of nodes in each layer is divided into equal-sized groups. For each task, we only keep a fixed number of active groups of nodes and nullify all the other nodes.

interpretable sub-networks.

2 Multi-task group dropout

2.1 Network architecture, groups and layers

Many architectures for multitask learning are based on a matching of subset of model parameters with tasks. Given the task and the input instance, only a subpart of the network will be involved in the computation of the output value, based on a predefined association between subnetworks and tasks. The adapter architecture of (Bapna and Firat, 2019) illustrates this strategy, where a task-dependent set of layers is activated for each task.

In our approach, we also require to know the task \( d \in \{0, \ldots, n_d - 1\} \) for each training and test instance. The structure of our Transformer networks (Vaswani et al., 2017) is however based on the notion of groups of nodes in the computation graph. At the input of each Transformer layer \( l \in \{1 \ldots L\}, \) we partition all input state vectors into \( n_p \) groups of nodes, and zero-out a task-dependent subset of these groups. The assignment of tasks to groups will be learned from the data, under the constraint that each task only activates exactly \( k \) groups of active nodes, while all the other values are nullified, akin to a dropout process (see Figure 1).

Formally, a group dropout mask \( m^d_l \) is a \( n_p \)-dimensional binary vector containing exactly \( k \) ones: group \( p \in \{0, \ldots, n_p - 1\} \) is retained for task \( d \) if \( m^d_l(p) = 1 \) and is dropped if \( m^d_l(p) = 0 \). We denote \( \Delta^\pi p = \{m \in \{0,1\}^{n_p} \mid |m|_{L_1} = k\} \) the set of all admissible masks, with \( |m|_{L_1} \) the \( L_1 \) norm of vector \( m \); \#\( \Delta^\pi p \) is the cardinal of \( \Delta^\pi p \).

Given \( m^d_l \), the mask \( r^d_l \) for task \( d \) in layer \( l \) is then derived as:

\[
r^d_l(i) = m^d_l(p) \quad \text{if} \quad p \times \frac{d_k}{n_p} \leq i < (p + 1) \times \frac{d_k}{n_p},
\]

where \( d_k \) is the dimension of the hidden state. The propagation of information within the network then depends on the current task value as follows:

\[
\forall l \in \{0, \ldots, L - 1\} : \tilde{h}^l = h^l \odot r^d_l,
\]

\[
h^{l+1} = \text{LAYER}^{l+1}(\tilde{h}^l),
\]

where LAYER\( ^l() \) represents all the computations in Transformer layer \( l \), \( \odot \) is element-wise product. It is applied at all positions of each layer in the encoder and in the decoder.

2.2 Training with latent dropout masks

Assuming standard notation for our translation model \( P(y|x, d; \theta) \) where \( y \) and \( \theta \) respectively refer to the input, output, and parameter vector, the latent variables \( m^d_l, l \in \{0, \ldots, L\}, d \in \{0, \ldots, n_d - 1\} \) are introduced as follows. We chose the prior distribution for \( m^d_l \) as the uniform distribution over \( \Delta^\pi p \):

\[
P(m^d_l|x, d; \theta) = \text{Unif}(\Delta^\pi p); \quad \text{variables for each layer are independent and collectively referred to as } m^d.
\]

For any (variational) distribution \( Q(m^1 \ldots m^{n_d}; \Phi) \) with parameters \( \Phi = \{\phi^1 \ldots \phi^{n_d}\} \), it is well-known that the log-likelihood is lower-bounded by the so-called ELBO function (hereafter denoted \( \ell \)), made of a summation of two terms: the distortion \( D \) and the rate \( R \) defined as follows:

\[
\ell(x, y, d; \theta, \Phi) = D(x, y, d; \theta, \Phi) - R(x, y, d; \theta, \Phi)
\]

(1)

\[
D(x, y, d; \theta, \Phi) = \mathbb{E}_{m^d \sim Q(m^d|x, d; \theta)} \log P(y|m^d, x, d; \theta)
\]

\[
R(x, y, d; \theta, \Phi) = KL(Q(m^d|x, d; \theta)||P(m^d|x, d; \theta)),
\]

where KL is the Kullback-Leibler divergence. We use \( -\ell(x, y, d; \theta, \Phi) \) as our surrogate training loss, as a tractable approximation of the likelihood, and try to minimize this function in \( \theta \) and \( \Phi \).

The variational distribution \( Q \) of \( m^d \) is defined independently on a layerwise basis; this means that each layer only involves a subset \( \Phi^d_l \) of variational parameters. \( Q \) is computed as follows:

\[
\text{Ind}^d = \{i_1, \ldots, i_k\} \sim \text{SRS(softmax(\Phi^d_l), k)}
\]

\[
m^d_l(i) = \mathbb{I}(i \in \text{Ind}^d),
\]

where SRS(\( \pi, k \)) denotes the process of sampling \( k \) times without replacement from the distribution \( \pi \), and \( \mathbb{I} \) is the indicator function. This modeling choice for the latent vector \( m^d \) is motivated by the Gumbel Top-K trick of Kool et al. (2019) that we
use below. Given our choices for the prior and the variational distributions, the two terms in Eq. (1) can be computed as:
\[ D(\ldots) = \mathbb{E}_{m^d \sim Q(m^d|d; \Phi)} \log P(y|m^d, x, d, \theta) \]
\[ = \mathbb{E}_{m^d \sim \tilde{Q}(G(0,1))} \left[ \log P(y|\tilde{m}^d, x, d, \theta) \right] \]
where the generation process \(G(0,1)\) is a product of independent Gumbel distributions, yielding:
\[ \forall d, g^d = [g^d_1, \ldots, g^d_{l_d}], \text{ with } g^d_l \in \mathbb{R}^{n_p} \]
\[ \forall p, g^d_i(p) \sim \text{Gumbel}(0, 1) \]
\[ \text{Ind}^d = \{i_1, \cdots, i_k\} = \text{Top-k} \{ g^d_i(0) + \Phi_{d}^i(0), \cdots, g^d_i(n_p-1) + \Phi_{d}^i(n_p-1) \} \]
\[ \tilde{m}^d_i(p) = \mathbb{I}(p \in \text{Ind}^d). \]
For the second term, the derivation is the following:
\[ R = \text{KL}(Q(m^d|d, \Phi)||P(m^d|x, d; \theta)), \]
\[ = - \sum_{l=1}^{L} \left( \mathbb{E} \left[ Q(m^d_l|d, \Phi) \right] - \log (#\Delta^d_{\text{min}}) \right) \]
\[ = - \sum_{l=1}^{L} \left( \mathbb{E} \left[ Q(i_1, \cdots, i_k|d, \Phi) \right] - \log (#\Delta^d_{\text{min}}) \right) \]
\[ \leq - \sum_{l=1}^{L} \left( \mathbb{E} \left[ Q(i_1|d, \Phi^d_l) \right] - \log (#\Delta^d_{\text{min}}) \right). \tag{3} \]
We prove inequality (3) in Appendix B. This inequality shows that an upperbound of \( R \) is \( \sum_{l=1}^{L} (\log (#\Delta^d_{\text{min}}) - \mathbb{H}(\text{softmax}(\Phi^d_l))) \) since \( i_1|\Phi^d_l \sim \text{softmax}(\Phi^d_l) \). During training, we thus maximize a sum over layers of the entropy \( \mathbb{H}(\text{softmax}(\Phi^d_l)) \) which performs a regularization over the parameters \( \Phi^d \) of the variational distribution.

Thanks to the Gumbel Top-K trick, we can move the parameters \( \Phi \) into the objective function and get rid of policy gradients, which have been reported to be very unstable (Kingma and Welling, 2014). However, the operator Top-k, which serves to define \( \tilde{m}^d_i \) in Equation (2), is not differentiable. Therefore, we approximate this function by the Soft-Top-K function defined as follows:
\[ \hat{m}^d_i(\tau) = \arg \min_{0 \leq m_i \leq 1; \tau \geq 0} - (g^d_i + \Phi^d_i)^T . m - \tau H_b(m) \]
\[ \forall 0 \leq m_i \leq 1, \quad 1^T . m = k \]
\[ \text{in which } \quad H_b(m) = - \sum_{i} m_i \log (m_i) + (1 - m_i) \log (1 - m_i). \tag{4} \]
In Appendix A, we prove that \( \lim_{\tau \to 0} \hat{m}^d_i(\tau) = \tilde{m}^d_i \). Furthermore, we also provide the computation of \( \tilde{m}^d_i(\tau) \) and prove that \( \tilde{m}^d_i(\tau) \) is a differentiable function w.r.t \( \Phi^d_l \) and that its gradients can be computed using the implicit differentiation theorem. During training, we approximate \( \hat{m}^d_i(\tau) \) by gradually decaying the hyper-parameter \( \tau \) to 0.2. The gradient of \( D \) w.r.t \( \Phi^d_l \) is computed using the chain rule as follows:
\[ \frac{\partial D}{\partial \Phi^d_l} = \frac{\partial D}{\partial \hat{m}^d_i(\tau)} \times \frac{\partial \hat{m}^d_i(\tau)}{\partial \Phi^d_l} \]
The gradient \( \frac{\partial D}{\partial \hat{m}^d_i(\tau)} \) is computed via autograd algorithm while \( \frac{\partial \hat{m}^d_i(\tau)}{\partial \Phi^d_l} \) is computed via implicit differentiation, as explained in Appendix A.

We jointly train the Transformer parameters \( \theta \) and the parameters of the variational distribution \( \Phi \) using the following multi-task loss.
\[ \mathcal{L}(\theta, \Phi) = \sum_{d=1}^{n_d} \mathbb{E}_{x \sim \mathcal{D}^d, y \sim \mathcal{T}^d(x)} [-\ell(x, y; d, \theta, \Phi)] \]
in which \( \mathcal{D}^d \) is distribution of task \( d \) over the input space \( \Omega^d \); \( \mathcal{T}^d : \Omega^d \to \Omega^d \) is the translation function for task \( d \), which our multi-task model needs to learn; \(-\ell(x, y, d, \theta, \Phi)\) is the ELBO loss, defined in Equation 1.

Finally, during inference, we define the dropout mask for layer \( l \) and task \( d \) as follows:
\[ \text{Ind}^d_l = \text{Top-k}(\Phi^d_l) \]
\[ m^d_l = \mathbb{I}(i \in \text{Ind}^d_l) \]
meaning that we simply pick the \( k \) most likely parameter groups for the task at hand, and define the state dropout mask accordingly.

3 Experimental settings

3.1 System design and configuration

3.1.1 Multidomain translation systems

Our systems for the multidomain experiments are designed as follows:

- **Transformer**: The embedding dimension for both encoder and decoder is set as 512, and the feedforward dimension is 2048; the multi-head attention mechanism contains 8 heads; 6 layers in the encoder; 6 layers in the decoder.
- **Adapter-based Transformer**: the intermediate feedforward dimension is set to 2048, as in Pham et al. (2021).
- **Transformer using Latent multi-task group dropout (LaMGD Transformers)**: There is no change in the architecture. We group the 512 nodes in each layer into 16 groups of 32 consecutive nodes. For each domain, only 12 out of the 16 groups are selected. The number
of parameters of the variational distribution is $L \times k \times L \times nd$, which is negligible in comparison to the size of the Transformer model.

- Transformer using heuristic multi-task group dropout (HMGD Transformer): we share 320 nodes for every task, and reserve 32 nodes for each task (totaling $320 + 32 \times 6 = 512$ nodes).

We set the dropout probability to 0.1. We train the multilingual Transformer model for 200k iterations with a batch size of 12k tokens using 4 V100 GPUs. The convergence of the standard Transformer is before 200k as its validation curve became flat near the 200k-th iteration. The LaMGD Transformer converged after 300k iterations with the same batch size. The convergence of LaMGD is controlled by its validation curve. Finally, we plug adapters to the multidomain Transformer model and finetune them for 25k iterations using the same batch size as the baseline.

### 3.1.2 Multilingual translation systems

The systems used in our multilingual experiments are implemented as follows:

- Multilingual Transformer: the embedding dimension for both encoder and decoder is set as 512, and the feedforward dimension is 1024, each multi-head attentions contains 8 heads as in (Wang et al., 2020a).
- Adapter based Transformer: the intermediate feedforward dimension is set as 128. We follow here the parameter setting of (Gong et al., 2021a).
- LaMGD Transformer: There is no change in the architecture. We group 512 nodes in each layer into 16 groups of 32 consecutive nodes. For each language, we select 12 groups.

We set the dropout probability to 0.3. We train the multilingual Transformer model for 40k iterations with a batch size of 960 tokens on 16 V100 GPUs as in Gong et al. (2021a). We train LaMGD Transformer for 50k iterations with the same batch size. The convergence of the models are controlled via their validation curves. Finally, we finetune the language-specific Adapters for 5k iterations.

All the translation systems are implemented with OpenNMT-tf\(^2\) (Klein et al., 2017).

### 3.1.3 Hyper-parameters

We choose $nd = 16$ so that the size of the dropout group is neither too small nor too large. The second important hyper-parameter in LaMGD is the number of selected groups in each layer, $k$, which we set to 12 in every experiments. By retaining 12/16 groups, we share on average 75% active groups between two domains or languages. This design ensures that the percentage of sharing is in the same ballpark as what we obtain with adapter modules. In our future work, we intend to analyze how these choices affect the final performance of the model.

The temperature parameter $\tau$ for the Soft-Top-K operator is gradually decreased from 0.5 to 0.2 according to the following policy:

$$\tau = \min\{0.2, 0.5 \times \exp^{-r \times \text{step}}\},$$

in which $r = 0.0001$. While Gong et al. (2021b,a) fixed $\tau$ to be 0.2, we select an anneal policy for $\tau$ proposed by previous studies (Jang et al., 2017). Finally, we set the weight of the entropy term to 0.0001 in the training loss in every experiments.

### 3.1.4 Latent variables initialization

We initialize the distribution of the latent variables uniformly. More precisely, we set $\Phi^d$, which generates the probability of the masks via the softmax activation function, to $0^{d}$.

### 3.2 Datasets and metrics

#### 3.2.1 Multidomain translation

We use the same data as in the recent work of Pham et al. (2021) on multidomain translation. The datasets\(^3\) for the multidomain translation experiments are detailed in Table 1. For each domain, the size of the dev set and the test set is 1 K.

#### 3.2.2 Multilingual translation

We evaluate our model on both one-to-many (O2M) and many-to-one (M2O) translation tasks borrowing the multilingual translation datasets from past studies. More precisely, we used:

- TED8-Related. Following the setting of Wang et al. (2020a), we use a subset of translations from Qi et al. (2018) between English and eight related languages.
- TED8-Diverse. The dataset consists of parallel sentences between English and eight diverse languages as in Wang et al. (2020a).

The languages used in the multilingual experiments are as follows (see statistics in Table 2):

- **Diverse set**: bos (Bosnian), Bulgarian (bul), French (fra), ell (Greek), hin (Hindi), Korean (kor) mkd (Macedonian), mar (Marathi);

\(^2\)https://github.com/OpenNMT/OpenNMT-tf

\(^3\)See https://github.com/qmpham/experiments/tree/main/tacl20
In Figure 4, we visualize the domains using their dropping masks concatenated and mapped to a 2d

<table>
<thead>
<tr>
<th>MED</th>
<th>LAW</th>
<th>BANK</th>
<th>IT</th>
<th>TALK</th>
<th>REL</th>
</tr>
</thead>
<tbody>
<tr>
<td># lines</td>
<td>2609 (0.68)</td>
<td>501 (0.13)</td>
<td>190 (0.05)</td>
<td>270 (0.07)</td>
<td>160 (0.04)</td>
</tr>
<tr>
<td># tokens</td>
<td>133 / 154</td>
<td>17.1 / 19.6</td>
<td>6.3 / 7.3</td>
<td>3.6 / 4.6</td>
<td>3.6 / 4.0</td>
</tr>
<tr>
<td># types</td>
<td>771 / 720</td>
<td>52.7 / 63.1</td>
<td>92.3 / 94.7</td>
<td>75.8 / 91.4</td>
<td>61.5 / 73.3</td>
</tr>
<tr>
<td># uniq</td>
<td>700 / 640</td>
<td>20.2 / 23.7</td>
<td>42.9 / 40.1</td>
<td>44.7 / 55.7</td>
<td>20.7 / 25.6</td>
</tr>
</tbody>
</table>

Table 1: Corpora statistics: number of parallel lines ($\times 10^3$) and proportion in the basic domain mixture (which does not include the NEWS domain), number of tokens in English and French ($\times 10^6$), number of types in English and French ($\times 10^3$), number of types that only appear in a given domain ($\times 10^3$).

- **Related set**: Azerbajiani (aze), Belarusian (bel), Czech (ces), Galician (glg), Portuguese (por), Russian (rus), Slovak (slk), Turkish (tur).

For all experiments, we report the BLEU score of Papineni et al. (2002) computed with SacreBLEU (Post, 2018). Statistical significance is computed with compare-mt⁴ (Neubig et al., 2019). We report significant differences at the level of $p = 0.05$.

4 Results and analyses

4.1 Multidomain translation

For these experiments, our main results are in Table 3, where we observe that the LaMGD Transformer achieves a significant improvement (+2.78) over the generic Transformer system with zero extra parameters. Moreover, LaMGD Transformer achieves performance that are equivalent on average to that of the Adapter systems, which is fine-tuned and contains approximately 25M additional parameters per domain. Variational mask learned from data by LaMGD also outperforms heuristic dropout mask HMGD by 0.5 in average.

4.1.1 Fuzzy domain separation

For this experiment, we reuse proposal of Pham et al. (2021), who measure the efficiency of a multidomain NMT system exploiting the proximity between domains. It uses the same data as in the previous experiment; however, the domain LAW is now randomly split into two pseudo-domains LAW₁ and LAW₂ of equal size. A truly multidomain system should be able to automatically detect the proximity between LAW₁ and LAW₂, and there should be no significant difference between the performance of a system trained with the six original domains (including LAW) or with the seven domains (including LAW by LAW₁ and LAW₂). Pham et al. (2021) reported a large gap between the two settings when using residual adapters. We replicated this setting and report the results obtained with the LaMGD Transformer system in Table 4.

The results in Table 4 show a performance decrease for the adapter-based system when training with two pseudo-domains LAW₁ and LAW₂. In contrast, the LaMGD model obtains very stable results. In Section 4.3, we show that our algorithm in fact computes the same sub-network for LAW₁ and LAW₂, that allows a full sharing of information between these two pseudo-domains.

4.2 Multilingual translation

Results for the multilingual experiments are in Table 5. The LaMGD Transformer achieves an improvement of 0.42, 0.33, 0.32 in average over the multilingual Transformer in the O2M-related, M2O-related, M2O-diverse conditions, respectively. Significant gains are observed for languages BEL, GLG (both direction), Hin and Bos (O2M direction) which are very low-resource languages in our sets. However, LaMGD Transformer is outperformed by the multilingual Transformer and language Adapters for the O2M-diverse condition.

4.3 Similarity between dropping masks

This section compares the sub-networks learnt for each domain or language pair by computing the average similarity between the corresponding dropout masks concatenated for all the layers of the underlying model. For the multidomain experiment, we analyze the case of pseud-domain separation reported in Section 4.1.1 in Figure 2a. We see that the sub-networks for LAW₁ and LAW₂ are identical, yielding a full sharing between the corresponding training sets. Furthermore, we observe a large distance between REL and the other domains, which is expected given that REL is quite distinct from the other domains. REL only share around 75% its active groups with other domains, as would be obtained by chance in our setting (see Section 3.1.3). In Figure 4, we visualize the domains using their dropping masks concatenated and mapped to a 2d

⁴https://github.com/neulab/compare-mt
Table 2: Data Statistics of TED8 Datasets

<table>
<thead>
<tr>
<th>Related</th>
<th>Diverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG</td>
<td>TRAIN</td>
</tr>
<tr>
<td>Azerbaijani</td>
<td>5.94k</td>
</tr>
<tr>
<td>Belarusian</td>
<td>4.51k</td>
</tr>
<tr>
<td>Galician</td>
<td>10.0k</td>
</tr>
<tr>
<td>Slovak</td>
<td>61.5k</td>
</tr>
<tr>
<td>Turkish</td>
<td>182k</td>
</tr>
<tr>
<td>Russian</td>
<td>208k</td>
</tr>
<tr>
<td>Portuguese</td>
<td>185k</td>
</tr>
<tr>
<td>Czech</td>
<td>103k</td>
</tr>
</tbody>
</table>

Table 3: Multi-domain translation. Boldface identifies best system for each domain.

<table>
<thead>
<tr>
<th>Model / Domain</th>
<th>LAW</th>
<th>IT</th>
<th>REL</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>40.3</td>
<td>59.5</td>
<td>49.8</td>
<td>80.0</td>
</tr>
<tr>
<td>HMGD Transformer</td>
<td>40.4</td>
<td>60.4</td>
<td>51.9</td>
<td>86.8</td>
</tr>
<tr>
<td>Adapter</td>
<td>39.5</td>
<td>61.0</td>
<td>53.1</td>
<td>91.0</td>
</tr>
<tr>
<td>LaMGD Transformer</td>
<td>40.3</td>
<td>60.4</td>
<td>52.4</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Table 4: Experiments with two similar pseudo-domains

<table>
<thead>
<tr>
<th>Model / Domain</th>
<th>O2M-related</th>
<th>M2O-related</th>
<th>O2M-diverse</th>
<th>M2O-diverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adapter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LaMGD Transformer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Multilingual Translation experiments. Boldface denotes significant gains over Transformer ($p = 0.05$).

For multilingual (TED-related) experiments, the training data contains four language families: (1) Turkic, with Azerbaijani and Turkish (AZE, TUR); (2) Slavic, with Belarusian and Russian (BEL, RUS); (3) Romance, with Galician and Portuguese (GLG, POR); and (4) Czech-Slovak, with Slovak and Czech (CES, SLK). We provide in Figure 2b the heatmap of the similarities between the dropout masks of our objective languages. We observe that each pair of languages in the same family correspond to brightest color except the di-
4.4 Ablation study

We discuss here the choice of the hyper-parameters $k$, the number of activated nodes in each layer, and its impact on the sharing level between the tasks. Table 7 shows the variance of performance when the number of activated nodes is changed, and the sharing level between tasks decreases in consequence. In addition, we also report in this section the effect of not choosing the number of groups $n_p$, which is assigned to 16 in the comparison of LaMGD and the contrasting methods. We show that setting $n_p$ to the layer’s size, which means the group size is 1, has a very similar performance as choosing $n_p$ heuristically.

<table>
<thead>
<tr>
<th>$k/n_p$</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 / 16</td>
<td>19.33</td>
</tr>
<tr>
<td>384 / 512</td>
<td>19.26</td>
</tr>
</tbody>
</table>

Table 7: Setting the size of group to 1 (o2m-related experiment). The quota of activated nodes is keep unchanged to 75%

5 Related work

Multidomain and multilingual translation systems have received considerable attention in the recent years, and an exhaustive survey is beyond the goal of this paper. Domain adaptation for neural MT is surveyed in (Chu et al., 2017), while multidomain MT systems are notably studied in (Saunders, 2021; Pham et al., 2021); for multilingual MT, the reader is referred eg. to (Chu and Dabre, 2018; Dabre et al., 2020). We focus on the most relevant subset of this literature below.

Language similarity The methods developed by (Sen et al., 2019; Kong et al., 2021) use language proximity to design parameter sharing strategies. The authors propose a multi-decoder model sharing the same encoder among languages and routing languages in different families to different decoders. These approaches share the same interest in expressing the proximity between tasks in the selection of task-specific parameters as our approach. However, our method learn the selection from a latent commonality in data instead of using a pre-defined selection such as "One language family per decoder" in (Kong et al., 2021).

Language-specific sub-networks. Frankle and Carbin (2019); Liu et al. (2019) study techniques to identify the most important parameters for the current task, so that masking the less important parameters during training does not hurt performance. Lin et al. (2021) adapts this idea for multilingual NMT, trying to identify language dependent subsets of parameters by pruning a fine-tuned model. Our approach also aims to map sub-networks to tasks: we do so by masking the output of each layer, rather than masking parameters. Furthermore, Lin et al. (2021) computes the masks via a heuristic selection; while our approach learns the masks with variational techniques.

Sparse Transformer The idea of adaptive sparsity is studied in several works. For instance, Li et al. (2020) propose to use a variable depth for dif-
different tasks. The authors aimed to match the depth of the sub-network to the complexity of the task. Gong et al. (2021b,a) also take an interest in the adaptive sparse Transformers, in which differ each task triggers the selection of specific heads in multi-head attention, layers, and blocks in feedforward matrices. Mixture-of-experts (MoE) constitute another effective approach to achieve sparsity. Using the Transformer architecture, the GShard model replaces a single feedforward (FFN) sub-layer with an MoE module consisting of multiple FFN sub-layers (Lepikhin et al., 2021; Fedus et al., 2021).

Adapter modules Adapters have proven to be very efficient for multi-task NLP (Houlsby et al., 2019; Bapna and Firat, 2019; Pham et al., 2020; Pfeiffer et al., 2020). In a nutshell, this technique consists in plugging several so-called adapter modules to the intermediate layers of a pretrained Transformer and finetuning these adapters on the downstream tasks. Adapters can also be trained without supervision for multilingual translation (Philip et al., 2020). However, the hard-coded separation between the domains of different tasks may lead to a catastrophic forgetting effect (Pfeiffer et al., 2021), which is a common problem in multi-task modeling using neural networks (McCloskey and Cohen, 1989). In multidomain translation, Pham et al. (2021) recently demonstrated the brittleness of adapters against fuzzy domain separations, out-of-domain distributions, and erroneous domain tags. Several subsequent studies have aimed to mitigate this weakness through a mixture of expert mechanism (e.g. (Pfeiffer et al., 2021)).

Zhang et al. (2021) propose to learn to route between shared and language-specific representations with a conditional language-specific routing while training the parameters of the underlying Transformer. This method is related to the Fusion-Adapters of Pfeiffer et al. (2021). Both approaches aim to select between shared and task-specific representations. The proximity between tasks is not taken into account in the routing mechanism. We propose a different approach to the problem of multi-task routing in the underlying network.

6 Conclusions and outlook

In this work, we have presented a novel method for multidomain and multilingual translation. It allows us to jointly search for an optimal assignment of sub-networks to tasks and to learn the parameters of the underlying network. Our method relies on a sound mathematical framework and an end-to-end optimization procedure; it only adds a small number of extra parameters. The additional training cost is also reasonable, amounting to 100k iterations in the multidomain setting, given the observed gains in performance. Experimentally, we achieve a large improvement over a Transformer baseline; our performance are also comparable to that of a strong a multi-task baseline using residual adapter modules which rely on a large number of extra parameters. For multilingual translation, our model outperforms multilingual Transformer and Language Adapters in 3 our of 4 settings. LAMGD seems specially beneficial for training languages with little parallel data, which can take advantage...
of the resources that are available for related languages. Besides, we provided a thorough analysis of the similarities between learned sub-networks and demonstrate a strong correlation between the learned similarities and the proximity of the corresponding tasks (domain or language).

There are several ways in which our methodology can be improved. In future work, we would first like to provide an complete variational framework to model both the number of groups, $k$, and the selection of the dropout masks. Second, we also intend to dispense with the domain information during inference: this would mean replacing the dependency on $d$ in the variational distribution by a dependency on the input $x$. Another interesting direction will be to consider adapting the size and capacity allocated to each domain / language, depending on the difficulty of the associated translation task. Addressing these questions will allow us replace heuristic choices in the architecture design with an increased dependency on the training data.

7 Ethical Considerations

MT technologies are generally intended to facilitate cross-lingual as well as cross-cultural communications. The methods presented here are notably interesting in the view to improve MT from and into English for low-resource languages, subject to the availability of data for a related language. We acknowledge that (a) our results should ultimately be backed-up large scale experiments involving much more languages – even though this goes against the idea of limiting the computing cost our experiments; (b) better architectures and training regimes can improve the translation quality for low-resource languages, yet will not solve the problem entirely. This means that additional work focusing specifically on developing resources for these languages should remain an important objective for future work.

8 Acknowledgements

This work was granted access to the Jean Zay HPC resources of [TGCC/CINES/IDRIS] under the allocation 2020-[AD011011270] made by GENCI (Grand Equipement National de Calcul Intensif).

References


Hongyu Gong, Yun Tang, J. Pino, and Xian Li. 2021b. Pay better attention to attention: Head selection in
multilingual and multi-domain sequence modeling. ArXiv, abs/2106.10840.


\[ \lim_{\tau \to 0} \hat{m}_d^l(\tau) = \tilde{m}_d^l. \]
We assume \( g^d_l(i_1) + \Phi^d_l(i_1) > g^d_l(i_2) + \Phi^d_l(i_2) > \cdots > g^d_l(i_{n_p}) + \Phi^d_l(i_{n_p}) \).

Because:

\[
\lim_{\tau \to 0} \sigma\left( \frac{g^d_l(i) + \Phi^d_l(i) + \nu}{\tau} \right) = \begin{cases} 
1, & \text{if } \tau > -(g^d_l(i) + \Phi^d_l(i)), \\
0, & \text{if } \tau < -(g^d_l(i) + \Phi^d_l(i)), \\
\frac{1}{2}, & \text{otherwise}
\end{cases}
\]

and

\[
\sum_{i=1}^{n_p} \sigma\left( \frac{g^d_l(i) + \Phi^d_l(i) + \nu}{\tau} \right) = k
\]

there exist \( \epsilon \) such that \( \forall \tau < \epsilon \), the solution \( \tilde{\nu} \) of (6) satisfies \( -(g^d_l(i_{k+1}) + \Phi^d_l(i_{k+1})) > \tilde{\nu} > -(g^d_l(i_k) + \Phi^d_l(i_k)) \). Furthermore, because sigmoid is monotonically increasing,

\[
\sigma\left( \frac{g^d_l(i) + \Phi^d_l(i) - (g^d_l(i_k) + \Phi^d_l(i_k))}{\tau} \right) < \sigma\left( \frac{g^d_l(i) + \Phi^d_l(i) - (g^d_l(i_{k+1}) + \Phi^d_l(i_{k+1}))}{\tau} \right)
\]

By taking the limit on both sides, we get the following results:

\[
\lim_{\tau \to 0} \tilde{m}^d_l(\tau)(i_u) = \begin{cases} 
1, & \text{if } u > k \\
0, & \text{if } u < k
\end{cases}
\]

And, because \( \sum_{i=1}^{n_p} \tilde{m}^d_l(\tau)(i_u) = k \), by taking the limit on both sides, we will have \( \lim_{\tau \to 0} \tilde{m}^d_l(\tau)(i_k) = 1 \). Finally, we have

\[
\lim_{\tau \to 0} \tilde{m}^d_l(\tau)(i_u) = \begin{cases} 
1, & \text{if } u \geq k \\
0, & \text{if } u < k
\end{cases}
\]

which is equivalent to \( \lim_{\tau \to 0} \tilde{m}^d_l(\tau) = \tilde{m}^d_l \).

B Appendix B

In this section, we give a simple proof of inequality (3). In fact, we only need to prove 
\[
\mathbb{H}[P(i_1, \ldots , i_k|\Phi^d_l)] \geq \mathbb{H}[P(i_1|\Phi^d_l)].
\]

The proof is as follows:

\[
= - \mathbb{E}_{i_1, \ldots , i_k, i_{k+1}} \left[ \sum_{j=k+1}^{k} \log P(i_{j}, \ldots , i_k|\Phi^d_l) + \log P(i_1|\Phi^d_l) \right]
\]

\[
\geq - \mathbb{E}_{i_1, \ldots , i_k, i_{k+1}} \left[ \log P(i_1|\Phi^d_l) \right]
\]

\[
= - \mathbb{E}_{i_1, i_{k+1}} \left[ \log P(i_1|\Phi^d_l) \right] = \mathbb{H}[P(i_1|\Phi^d_l)]
\]

C Appendix C

Algorithm 1 Training LaMGD

 Require:
\begin{itemize}
  \item \( n_d \) corpora \( C^d, d \in [1, \ldots , n_d] \) for \( n_d \) domains equipped by an empirical distribution \( D_d(x) \)
  \item number of groups: \( n_p \); number of retained groups: \( k \)
  \item \( i = 0; \text{iter} \_\text{num} \)
\end{itemize}

1: repeat
2: Pick a batch from domain \( d \)
3: Sample \( \forall l, \forall p : g^d_l(p) \sim \text{Gumbel}(0,1) \)
4: Solve the equation \( \forall l \)
\[
\sum_{i=1}^{n_p} \sigma\left( \frac{g^d_l(i) + \Phi^d_l(i) + \tilde{\nu}}{\tau} \right) = k
\]

using binary search

Compute mask of each layer \( \forall l \)
\[
\tilde{m}^d_l(\tau) = \sigma\left( \frac{g^d_l + \Phi^d_l + \tilde{\nu}}{\tau} \right)
\]

Apply masks to their corresponding layer \( \forall l \in [0, \ldots , L-1] : h^l = h^l \odot r^d_l \)
\[
h^{l+1} = \text{LAYER}^{l+1}(h^l)
\]

D Appendix D

Figure 4: Visualization of domains according to their dropout masks (a large vector concatenating the dropping masks of all the layers of the model) constructed by PCA.
7: Compute gradient of training loss over the underlying Transformer
\[ \Delta \theta = \frac{\partial L}{\partial \theta} \]

8: Compute gradient over the Soft-Top-K masks
\[ \frac{\partial D}{\partial \hat{m}_d(\tau)} \]

9: Compute implicit gradient of the Soft-Top-K masks over \( \Phi_d \)
\[ \frac{\partial \bar{v}}{\partial \Phi_d} = -\left( \frac{\partial g}{\partial \bar{v}} \right)^{-1} \times \frac{\partial g}{\partial \Phi_d} \]
\[ \frac{\partial \hat{m}_d(\tau)}{\partial \Phi_d} = \frac{\partial \hat{m}_d(\tau)}{\partial \Phi_d} \times \frac{\partial \bar{v}}{\partial \Phi_d} + \frac{1}{\tau} \exp\left( \frac{g_d(i) + \Phi_d(i) + v}{\tau} \right) \]
\[ \frac{\partial \hat{m}_d(\tau)}{\partial \Phi_d} = \frac{\partial \hat{m}_d(\tau)}{\partial \Phi_d} \times \frac{\partial \bar{v}}{\partial \Phi_d} + \frac{1}{\tau} \frac{\exp\left( \frac{g_d(i) + \Phi_d(i) + v}{\tau} \right)}{1 + \exp\left( \frac{g_d(i) + \Phi_d(i) + v}{\tau} \right)^2} \]

10: Compute the gradient the training over \( \Phi_d \)
\[ \Delta \Phi_d = \frac{\partial D}{\partial \hat{m}_d(\tau)} \times \frac{\partial \hat{m}_d(\tau)}{\partial \Phi_d} + \frac{\partial \text{softmax}(\Phi_d)}{\partial \Phi_d} \]

11: Update \( \theta \) and \( \Phi_d \) with their gradients
12: \( i = i + 1 \)
13: \textbf{until} \( i > \text{iter\_num} \)