CS546 Class Project: Multi-task learning in NLP

Abhinav Kohar (aa18), Sai Krishna Bollam (sbollam2), Shivank Mishra (smishr25), Sidhartha Satapathy (ss46)

May 3, 2018

1 Introduction

Dependency between related tasks is an area of active research in Natural Language Processing. Recently, a large number of tasks under the umbrella of Multi-Task Learning (MTL) have been done, which aim to leverage useful information contained in multiple related tasks, to help improve the generalization performance of all the tasks. The idea stems from the usefulness of knowledge learned from multiple tasks for a complex task.

From a Machine Learning perspective, the tasks in MTL setup provide inductive bias to other tasks, improving the generalization of all the tasks. Our primary focus is Multi-task learning for Neural Machine Translation, with other auxiliary tasks. Specifically, we used source and target language auto-encoders, POS tagging model as the auxiliary tasks, and did one to many translations from English to Vietnamese and German where Vietnamese translation serves as an auxiliary task. *All these tasks have been coded by us and code is shared on gitlab Abhinav et al. (2018).*

2 Multi-task Learning

One can share representations between related tasks and and help any model generalize better on the task of interest. This way of getting a better model is called Multi Task Learning (MTL). In case of deep learning, there are primarily two different ways of performing MTL, based on how the parameters are shared between the tasks.

2.1 Hard Parameter Sharing

In this model, some or all of the parameters of the network are shared between multiple tasks. In general, there can be a set of layers in the network that are common across the tasks and multiple task-specific layers. So while this model provides enough regularization as a result of sharing some parameters, the task specific layers cater to specific tasks with possibly richer features.

2.2 Soft Parameter Sharing

Each of the tasks in soft parameter sharing models have their own parameters. The parameters are constrained from drifting too far from each other by including some regularization on the difference between the parameters.

Additionally, MTL has been observed to improve performance across tasks as a result of data augmentation, attention improvement by learning from noisy data, eavesdropping on features from other tasks, bias in representations across tasks and regularization, Ruder (2017).

3 Background

A lot of methods have been developed for MTL using deep learning. Discussed below are a few important networks from literature.

Long and Wang (2015) designed a deep relationship network to improve upon models in MTL for computer vision, which share convolutional layers, but learned the task specific fully connected layers. This works well for well studied problems in computer vision, but it has issues with novel tasks.

The CNN models learn transferable features that can adapt to multiple tasks. However, the challenge in the experiment was that in MTL each task is provided with a limited amount of labeled data, which is not enough to provide reliable classifiers without over-fitting. Therefore, a Multi-linear Relationship Network (MRN) was designed to model the task relationships. Each pair of tasks can help enable knowledge transfer if the tasks are related, and help remove negative transfer if they are unrelated.

The MRN model was built upon AlexNet, which is comprised of 5 convolutional layers and 2 fully-connected layers.

$$\mathbf{h}_n^{t,\ell} = a^\ell \left(\mathbf{W}^{t,\ell}\mathbf{h}_n^{t,\ell-1} + \mathbf{b}^{t,\ell}\right)$$

The fully connected layer learns a nonlinear mapping for task t, where $h_n^{t,l}$ is the hidden representation of each point $x_n^{t,l}$, $w^{t,l}$,and $b^{t,l}$ are the weight and bias parameters, and a^l is the activation function, taken as ReLU a^l(x) = max(0, x) for hidden layers or softmax units $a^l(x) = e^x / \sum_{j=1}^{|x|} e^{x_j}$ for the output layer.

The model shares the network parameters of the layers across the different tasks, without explicitly modeling relationships of features of variety of tasks. The layers also utilized pre-trained model from ImageNet 2012, and fine-tuned conv1-conv5 layers.

This paper also raised an important point that the fully connected layers are geared towards their original task, while compromising on the the target task, therefore it deteriorates the multi-task learning. Another point that paper raises is that MTL is designed for binary classifications tasks, which may not work well for deep networks since they adopt multi-class softmax regression.

Their algorithm is an optimization problem which is jointly non-convex with respect to the parameter tensors W as well as feature covariance $\sum_1^l$, class covariance $\sum_2^l$, and task covariance $\sum_3^l$

The setup of the experiment utilizes:

- Office-Caltech dataset, which is the benchmark for multi-task learning and transfer learning , it contains 4,652 images from Amazon, Webcam, and CSLR.
- ImageCLEF dataset that contains 12 common categories from public datasets of Caltech-256, ImageNet 2012, Pascal VOC 2102, and Bing.
- Office-home, which is used to evaluate transfer learning.

Deep multi-task learning method DMTL-TF outperforms shallow multi-task learning methods with deep features as input, which confirms the importance of learning deep transferable features to enable knowledge transfer across tasks. The paper however did not provide enough insight into how to explore task relationships of multi-class classification MTL.

Lu et al. (2016) develop a bottom up approach that begins with a thin network and later they dynamically widen it, using a criteria in training phase that leads to grouping of related tasks. Since the method is greedy, an optimal model might not be found, making it hard to learn complex interactions between tasks.

The authors have aimed for automatic learning of multi-task architecture, based on branching, and selective sharing for tasks, with automated learning of kind of tasks to share.

The model also required low memory footprint and low latency when the forward pass was performed through the network. It is initialized from a pre-trained VGG-16 model by selecting a subset of filters that used simultaneous orthogonal matching pursuit (SOMP). The SOMP produced approximate results to initialize the parameter matrix.

The experiment used CelebA dataset and Deepfashion dataset. The model was evaluated using standard classificaton accuracy and top-10 recall rate. The model was evaluated for top-3 and top-5 classification accuracy, to compare with the benchmark results.

They used 3 baselines. The VGG-16 model was trained on imdb-wiki gender classification. The second baseline was a low rank model with low rank factorization at all layers. The third is a thin model initialized using SOMP method.

The experiment started with a thin neural network and expanded during training by a multi-round branching mechanism. This helped them understand the tasks that share the features. The setup helped the researchers achieve a thin baseline that is 6 times faster, 500 times more compact than the VGG-16 baseline, but still quite accurate.

Just as in soft parameter sharing, Misra et al. (2016) propose a cross-stitch network, which has two different model architectures. Here they use "cross-stitch" layers to help task-specific network use the knowledge of the related task.

The cross-stitch units in this paper are generalized way of learning shared representations for MTL in CNN. The cross-stitch model is represented as linear combination. They have utilized end-to-end CNN. The CNN units generalize over several multi-task architectures, eliminating the need to search over several multi-task architectures on per task basis.

The experiments presented in this paper show cross-stitch method brought considerable gains to MTL.

The split architecture utilized in this paper gives space for a wide variety of shared and task-specific representations. The cross stitch units are utilized in a network to combine two or more tasks. At every layer, the representation sharing is modeled by learning the linear combination of activation/feature maps.

The cross-stitch units try to find the best shared representations for multi-task learning. The shared representations are modeled using linear combinations, and learn the optimal setup of linear combinations for a given set of tasks. In the experiments they have performed ablation to understand the units and training procedure. For ablative analysis they have considered semantic segmentation(SemSeg) and Surface Normal Prediction on NYU-v2 dataset.

In order to setup the the cross-stitching, they have combined two AlexNet architectures using the cross-stitch units. The cross stitch units are applied on activation maps for different pooling, and fully connected layers.

They experiments with cross-stitch networks for two pairs of tasks: semantic segmentation and surface normal prediction on NYU-v2 and object detection and attribute prediction on PASCAL VOC 2008.

Training was performed using Fast-RCNN using a 21 way 1-vs-all classification with 20 foreground and 1 background class. The setup requires Fast-RCNN construction of mini-batch of 1:3 foreground to background ratio, to prevent data imbalance between the classes.

Their proposed cross-stitched network composed of two sub-networks improve results over baseline one-task network and the ensemble. The ensemble has twice the parameters, but the cross-stitch method performs better.

On the other hand, people try to find better task hierarchies for MTL in Natural Language Processing (NLP). Søgaard and Goldberg (2016) show that when using low level NLP tasks like part of speech tagging and named entity recognition as auxiliary tasks, they should be supervised at lower layers in the network.

Their model uses a multi-task learning architecture with deep bi-directional RNNs with task supervision, by sharing one or more bi-RNNs. In the paper they have attempted to show that low level tasks are modeled better at lower layers.

The MTL architecture is used for sequence tagging with deep bi-RNNs.

The experiment uses depp bi-RNNs: RNN_F and RNN_R, where one RNN reads in forwards and another in reverse.

In the experiment, they tried POS-tagging, syntactic chunking, and CCG supertagging.

While performing in-domain MTL, POS, Chunking, and CCG data are form English Penn Treebank.

The MTL training is done for either (POS+chunking) or (POS+CCG), with POS being the lower-level task. The experiment included three architectures: single task training for higher-level tasks without the POS layer, MTL with both tasks feeding off of the outer layer, and MTL where POS feeds off of the inner (1st) layer and the higher-level task on the outer (3rd) layer

The results obtained from CHUNKS are very close to the state-of-art. The results obtained from the MTL model are significantly better than their own baseline, and POS supervision at the lower layer is consistently better than standard MTL.

All these tasks above have been trying to learn the structure of sharing. But, a refreshing approach is adjusting the weights of each task in a multi-task loss function. This loss function tries to maximize Gaussian likelihood with task dependent-uncertainty. Kendall et al. (2017) develops this weighted multi-task loss function approach.

In this paper model utilizes multi-task learning with multiple regression and classification. The loss function weights for multiple tasks are difficult to set manually. The paper uses homeoscedastic task uncertainty to principled multi-task loss to simultaneously learn various classification and regression losses. The architecture mentioned is unified for semantic segmentation, instance segmentation, and depth regression. The paper also showed the importance of loss weighting in deep learning and a way to obtain better results than separately trained models.

The model used homoscedastic noise term in order to optimize the task weights which helps keep the weights dynamic while training.

The architecture of the paper split into:

Semantic Segmentation: The cross-entropy loss is used to learn pixel-wise class probabilities, the loss is averaged over pixels with semantic labels in each mini-batch.

Instance Segmentation: This method was utilized to define which instance a pixel belongs, to find an association between a pixel and instance's centroid.

Depth Regression: They also trained with the supervised labels using pixel-wise metric to get inverse depth using a L_1 loss function. Such an inverse depth estimation helps in estimating points at an infinite distance.

The method utilized CityScapes dataset for road scene understanding.

The results they obtained showed that correctly weighing loss terms is important for MTL problems. The paper gave results that showed that homoscedastic uncertainty is to obtain weight losses. They also showed that a principled loss function can improve performance for scene understanding with a unified architecture.

Work by (Dong et al., 2015) generalizes machine translation to sequence-to-sequence modeling, for concurrently translating sentences from one source language to multiple target languages. This work proposes a framework which shares source language representation and separately models different target language translation. This framework has shown to work for both bigger and smaller datasets. Also, the translation accuracy for both datasets is better than individually learned model for machine translation.

The model uses unified neural machine translation as a sequencing problem. It uses a multi-task framework that is shared across different language pairs, and every target language has its own decoder.

They did two groups of experiments to show the effectiveness of their framework. The aim of the first experiment is to prove that multi-task learning helps to improve translation performance given enough training data for all language pairs. In the second experiment, they proved that for some resource-poor language pairs with a few parallel training data, the language translation performance could be improved as well.

In all there are 3 experiments for these two groups. In the first experiment they used the multi-task learning (MTL) model trained separately on each parallel corpora.

The second experiment utilized the same setup as the first. However, they mimic the situation where some parallel corpora are resource-poor and maintain only 15% data on two parallel training corpora.

In the third experiment they tested the model learned from previous two experiments on the WMT 2013 dataset.

The dataset used in this experiment was Europarl, which is a corpus of 21 languages. The source language and target used, consisted of 30k of most frequent words.

The results show the BLEU scores on case-sensitive dataset. Only with 15% of the parallel training corpus of English-Dutch, and English-Portuguese, the model was able to improve the performance on all target languages, eg. languages that have Latin, and Germanic roots. The three experiments showed positive results. The framework is able to reduce the issue of data scarcity in translation for languages that have less resources. The proposed model is efficient and gets faster and better convergence for both resource-rich and resource-poor language pair under the multi-task learning.

In the work by Luong et al. (2015a), the experiments are centered around translation tasks, where the motivations is to find what other tasks are useful in translation.

Multiple multi-task learning scenarios have been explored by combining different size tasks. In first setting they combine a large task (machine translation) with: (a) a small task Penn Tree Bank (PTB) parsing. In another setting a medium-sized task image caption generation is combined with machine translation. Additionally, another large task parsing on the high-confidence (HC) corpus is mixed machine translation. In the end, unsupervised tasks, like auto-encoders and skip-thought vectors are merged with machine translation. In terms of evaluation metrics, validation and test perplexities have been reported for all tasks. Additionally, test BLEU scores are calculated for the translation task to see generation accuracy.

Linguistic resources can be used for neural machine translation by multi-task learning, Niehues and Cho (2017). Three important decision aspects are introduced: pairs of tasks used for training, training schedule and how much parameters are shared, which of course depends on network architecture.

This work tried to integrate additional linguistic resources into neural model. They have used an attention based sequence-to-sequence model for all the tasks.

MTL was performed for three different tasks from German to English, German fine grained POS tagging and German NE tagging. The experiment involves sparse dataset to show usefulness of MTL, by using only German to English TED data for translation task.

The data used was 4M tokens of WIT corpus for German to English training data. The POS tagger was trained on 720K tokens the Tiger Corpus, so there is lot of out-of-domain data. The encoder uses word embeddings of size 256 and bi-LSTM with 256 hidden layers for each direction. They have used multi-layer perceptron with 512 hidden units and tanh activation function. Meanwhile the decoder is conditional GRU units with 512 hidden units. The models were trained with Adam optimization.

The output was evaluated with BLEU, BEER, and the CharacTER. They provide results of the baseline neural MT system trained on parallel data. They also evaluated models trained on translation and POS tagging. Even though POS data is out-of-domain, there was a big improvement an all the three architectures, consistently across three metrics.

Results show that multi-task learning improves the translation up to 1 BLEU points and 1.5 characTER points (reduction). As a by product, they were also able to improved the performance of the POS tagging by 30% to 50% relatively. This would be helpful since data annotation for many NLP tasks is time consuming and expensive.

Paper by Dalvi et al. (2017) tries to understand and improve morphological learning in Neural machine translation decoder. It is analyzed how much morphology an NMT decoder learns. It is also studied whether addition of target language morphology tags leads to better translation. This model is jointly trained as multi-task learning problem.

In the end, the work by Ruder et al. (2017) proposes how to generalize deep learning approaches for hard parameter sharing architectures, ”cross-stitch”, block sparse regularization approaches, NLP approaches that create task hierarchy. Basically, the model tries to tell what layers and subspaces are shareable and what are best ways to represent input sequences. The model is generalization of various MTL and transfer learning algorithms. The authors studied which task properties predict gains and which properties correlate with specific types of sharing.

When using deep learning neural networks it hard to predict if sharing of knowledge of loosely related tasks will lead to better results. In order overcome this issue the model introduces Sluice Networks, which is a framework for multi-task learning where sharing is controlled by number trainable parameters.

The experiment is performed on three task pairs, for seven different domains by utilizing data from OntoNotes 5.0, and achieving up to 15% average error reductions

In the main task, they have used chunking, named entity recognition, and a simplified version of semantic role labeling where they have identified headwords(to minimize the preprocessing of SRL , and then paired them with part-of-speech tagging (POS) as an auxiliary task.

The authors used state-of-the-art BiLSTM-based sequence labeling model. At every time step the BiLSTM model (3 layers of 100 hidden dimensions), received an input of a concatenation of 64-dimensional word embedding, over 100-dimensional character embeddings. Both word and character embeddings are randomly initialized. The output layer is an MLP with a dimensionality of 100.

The model used Stochastic Gradient Descent at an initial learning rate of 0.1, and a learning rate decay.

The models were evaluated on both in-domain, and out-of-domain test to evaluate the out-of-domain generalization ability.

The baseline models were compared with the single-task model training on chunking(CHUNKS), low supervision model that predicts auxiliary task at first layer , cross-stitch networks, and MTL model based on hard parameter sharing.

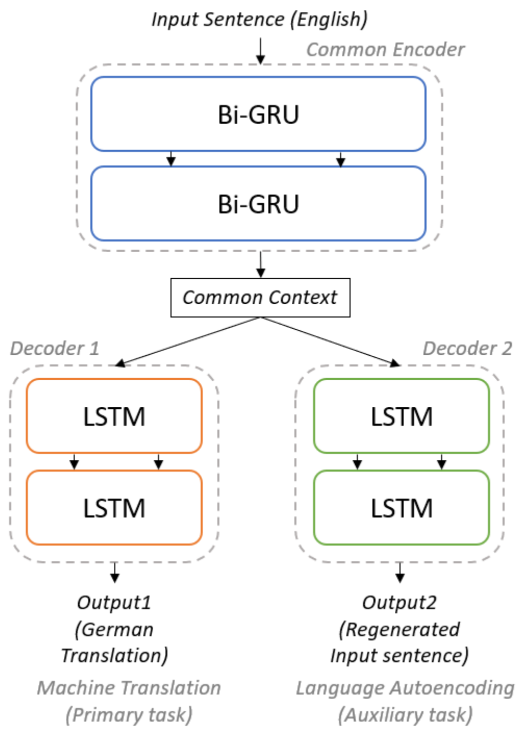
The results show that the authors found the sluice networks significantly outperformed all the model architectures. The single task learning was most useful in domain setting, where the test and train data was the same. However, on out-of-domain data, hard parameter sharing performs better. Low supervision model performed slightly better. Cross-stitch networks provided some improvement. The sluice networks gave the best performance among all the models.

4 Data and evaluation

We used the translation portion of multi-modal task in WMT16 dataset, which was a part of the first conference on Machine Translation (WMT16), wmt (2016). The source and target languages are English and German respectively. This parallel corpus has a total of 29,000 sentence pairs in training set, 1014 and 1000 pairs in validation and test sets respectively. There are a total of 9694 unique words in the source language and 17631 in target language. For one of the experiments, we also use the English-Vietnamese parallel corpus, which contains 113,618 sentence pairs with 11,667 words in Vietnamese vocabulary.

We evaluated translation performance of different multi-task architectures by comparing them with the models trained only for translation. We report the performance of models on BLEU and METEOR 1.5 metrics. The BLEU score is cumulative of all n-gram overlaps for $n \in [1,4]$. We also applied brevity penalty on this. As our primary task is machine translation, we did not specifically report the performance of auxiliary tasks in comparison to their respective baseline models. However, in a setting where we are training both the models to benefit from each other, it would be appropriate to report such scores. The detailed analysis of auxiliary tasks provide scope for future work.

Figure 1: Shared Encoder and Separate Decoders


Input Sentence (English)
Common Encoder
Bi-GRU
Bi-GRU
Common Context
Decoder 1
Decoder 2
LSTM
LSTM
LSTM
LSTM
Output1 (German Translation)
Machine Translation (Primary task)
Output2 (Regenerated Input sentence)
Language Autoencoding (Auxiliary task)

5 Experiments

In this section we present some of the many architectures and respective tasks that we've experimented on. We present the results for each of the experiments separately, as the baseline codes changed according to the multi-task architectures used. So, comparing to the right baselines would provide a better sense of how the models work.

5.1 En-Ge Translation and English Auto-encoding

5.1.1 Architecture

In this experiment, we used the hard parameter sharing model with a shared encoder and separate decoders for each of the tasks. The architecture is similar to the one shown in Figure 1. For training, we use bidirectional RNNs with Gated Recurrent Units and a 2 layer stack on the encoder side. Each of the decoder has a 2 layer RNN with long short term memory and are unidirectional. The attention mechanism used is the Key-Value attention Vaswani et al. (2017). Key-Value attention is an interpretation of attention mechanism in which the hidden state outputs from every time-step are considered as *key-value* pairs. On the decoder side, the current value of input is considered to be a *query*, and it's match with the keys is considered as the attention at that key's time step. We also tried using attention mechanism mentioned in Luong et al. (2015b) and Bahdanau et al. (2014). However, our experiments were mostly using the key-value attention mechanism and we present only those results here.

Table 1: Single Encoder-Multiple Decoders. Translation with English Auto-encoding

Machine Translation Results for German		
Models	**METEOR 1.5**	**BLEU**
Without Auto-encoder	46.91	26.35
With Auto-encoder	47.95	26.82
Improvement	**+1.04**	**+0.47**

5.1.2 Training

The recurrent units have a hidden size of 512 units on encoder and both the decoders. Embedding of dimension 512 is used on both source and target dimensions. Also, for the source embedding, the weights are shared with the second decoder, which does the English language auto-encoding. For the key-value attention, the dimension for keys and values are set at 256 units. A vertical dropout of 0.2 and on embedding was used. The initial cell and hidden state values of the decoders are parameter that are learned during training.

All the words that appear less than twice are removed from the dataset for this part, removing sentences containing them. The data contained about 20,064 sentence pairs for training, 596 pairs for validation and 526 pairs for testing. There are a total of about 5864 words in the source language and 7700 words in the target language.

The training is done with SGD optimizer with a batch size of 16 and a learning rate of 0.001, decayed to half on every epoch after 4 epochs. Same learning rate was used across each of the encoder and decoder modules. The decoder for primary task was trained 85% of the times, and the auxiliary task decoder 15% of the time, switching between batches. The loss calculated on each decoder is back-propagated only through that particular decoder and the common encoder. Training was suspended when the BLEU score on validation stabilized after 13 epochs of training (9 epochs in case of single task training for translation). The results should be reproducible from the code[1].

5.1.3 Results

The multi-task model performs slightly better than the single task model as shown in table 1. Also, the baseline BLEU here is different from the subsequent sections, as the training models are different for a fair comparison. Moreover, no beam search was used in the results shown here.

5.1.4 Observations

The multi-task model appears to perform much better than the single task translation model with same number of parameters in encoder and decoder used for translation only. The model performance seems to vary a lot with the dimension size used for embedding. This was verified by varying the embedding size, while keeping all other parameters such as hidden dimension and layers constant. As word embedding requires a lot of training to effectively represent the data in low dimensional spatial form, the training length used for translations isn't sufficient. So using a very high dimensional embedding might result in the embeddings not being good enough, so we need to strike a correct balance for the size of the word embeddings to obtain better results.

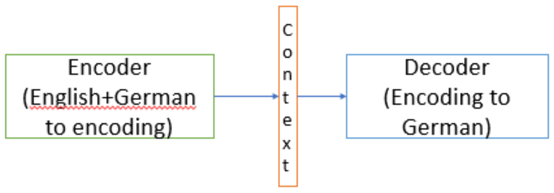
Here are samples of translations of this model, one where the translation is perfect and other where it was bad, shown in table 2:

[1]https://gitlab.engr.illinois.edu/ss46/MLinNLProject/blob/one_to_many/Final.py. Although it is in the OpenNMT folder, it isn't linked to OpenNMT and is written from scratch

Table 2: Multitask German Translation results: English Auto-encoding

Machine Translation examples for German
Source sentence 1: girls in red costumes are dancing .
Target reference: mdchen in roten kostmen tanzen .
Model output: mdchen in roten kostmen tanzen .
Source sentence 2: a butcher , with whole animals hanging in his storefront .
Target reference: ein metzger , in dessen geschft ganze tiere hngen
Model output: ein rollt mit einer brettspiel in der stock .

Figure 2: Complete Parameter Sharing Model


Encoder (English+German to encoding)
Context
Decoder (Encoding to German)

The quality of translations, in terms number of sentences translated perfectly, was higher in the multi-task model compared to single task.

5.2 En-Ge Translation and German Auto-encoding

5.2.1 Architecture

We have incorporated multi-task learning into the OpenNMT Klein et al. (2017) model as one of our models. The Multi-task idea over here is to improve the primary task of machine translation with German auto-encoding as the auxiliary task. The architecture demonstrates complete parameter sharing where we have one encoder and one decoder, 2. The model parameters are as follows, we train new word embeddings for the purpose of this task and use the word embedding size as 50. In addition, we would like to say that the size of the hidden vector we use is equal to 500. Furthermore, we use LSTM as the decoder as well as the encoder and the decoder uses Luong global attention. Finally, we would like to add that we learn embeddings for German and English in a shared space on the encoder side. For the purpose of training the decoder for the auto-encoding task we input a German sentence to the encoder and for the purpose of machine translation we input an English sentence.

5.2.2 Training

For the purpose of training this model, we limit the maximum length of the sentences to 50. The dropout is set to 0.3 for both the encoder and the decoder with a learning rate set to 1 and the learning rate decay set to 0.5. We train the model for about 15 epochs each for the auto-encoding task and for the machine translation task and use the SGD as the optimizer. We talk about obtained results obtained in the next section.

Table 3: Shared Encoder and Decoder

Machine Translation Results for German		
Models	**METEOR 1.5**	**BLEU**
Without auto-encoder	47.68	28.52
With auto-encoder	48.03	28.94
Improvement	**+0.35**	**+0.42**

5.2.3 Results

The multi-task model again performed slightly better than the single task model as shown in table 3. As shown in the table we obtain an improvement on both METEOR and BLEU score through our models.

5.2.4 Observations

We observe that the model performance varies a lot with the dimension size used for embedding and we need to find an optimal embedding size.

5.3 En-Ge Translation and German POS Tagging

5.3.1 Architecture

In this experiment we use the idea of hard parameter sharing model, where most of the parameter are shared however the final layers are task specific, since it worked better than the other sharing models. The architecture is shown in Figure 3. The attempt over here is to improve the machine translation using POS tagging as the auxiliary task. The model parameters are similar to the previous sections, we find that the LSTM work very well as the encoder and decoder and use the same for this experiment, which is different from previous experiments. We use the hidden vector size as 500 and train the word embedding size as 50. Furthermore, we use Luong global attention. Finally, we would like to add that we have multiple loss functions for the two tasks and try to optimize them.

So, we have two output layers in parallel, first to predict the target word, and second to predict it's the POS tag. Both the output layers have their own loss functions. While in training we combine losses from both output layers to jointly train the system. In this experiment, the training data is multi-target corpus from source to target words and from source to target POS tags. Here the number of outputs is 2 (N=2). So, given a set of training examples $D = (s^{(n)}, t^{(n)}, pos^{(n)})_{n=1}^{N}$; where s is the source sentence, t is the target sentence, pos is the POS tag sequence, the new objective function to maximize is :

$$\mathscr{L} = (1-\lambda)\sum_{n=1}^{N} logP(t^{(n)}|s^{(n)};\theta) + \lambda \sum_{n=1}^{N} logP(pos^{(n)}|s^{(n)};\theta)$$

Here λ is the hyper-parameter to decide whether to focus on German translation or POS tagging. We have shown results for $\lambda = 0.2$ giving best results for machine translation.

For the purpose of training the machine translation task the output from the decoder is mapped to the final output layer of size equal to the German vocabulary. For the purpose of POS tagging the output from the decoder is mapped to the final output layer of the size of the number of POS tags. Note: The POS tags were generated using Stanford POS-tagger.

Figure 3: Parameter Sharing with task specific layers

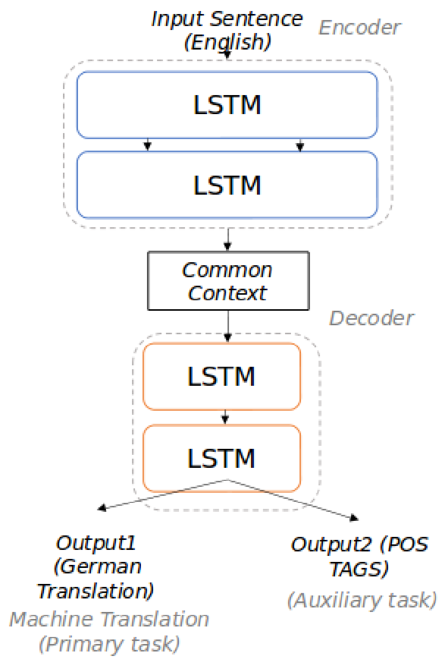


Table 4: Single Encoder and Single Decoder with separate final layers for Machine translation and POS Tagging

Machine Translation Results for German		
Models	**METEOR 1.5**	**BLEU**
Without POS Tagging	46.91	26.35
With POS Tagging	47.43	25.91
Improvement	**+0.52**	**-0.44**

5.3.2 Training

While training our models, we limit the maximum length of the sentences to 50. In this case, the dropout is set to 0.4 for the encoder and 0.3 for the decoder with a learning rate set to 1 and the learning rate decay set to 0.5. We train the models together for 13 epochs each for the machine translation task and POS tagging task and use the SGD as the optimizer. We talk about obtain the results obtained in the next section.

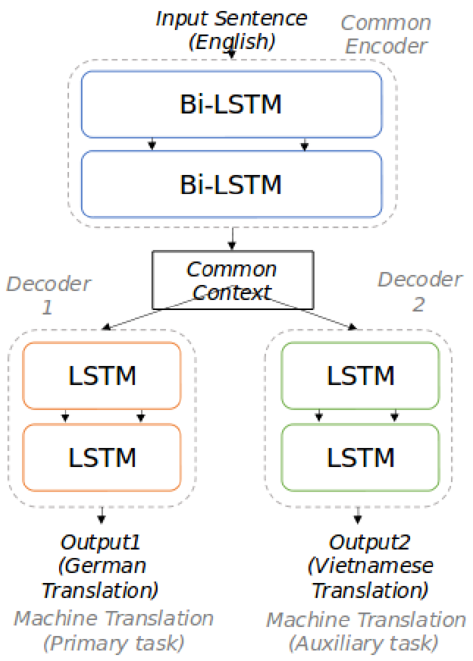
5.3.3 Results

The multi-task model performs better with METEOR as the metric, however perform worse for the BLEU score as shown in table 4.

5.3.4 Observations

The major observation we have for this experiment is that the results reduced for the BLEU-score when we use POS tagging is the auxiliary task, this could be because of the fact that the POS tags task tries to learn sparse representations in contrast to the machine translations task which learns dense representations.

Figure 4: Parameter Sharing on the encoder with different decoders for Vietnamese and German



Input Sentence (English)
Common Encoder
Bi-LSTM
Bi-LSTM
Common Context
Decoder 1
Decoder 2
LSTM
LSTM
LSTM
LSTM
Output1 (German Translation)
Output2 (Vietnamese Translation)
Machine Translation (Primary task)
Machine Translation (Auxiliary task)

5.4 En-Ge Translation and En-Vi Translation (One-to-Many System)

5.4.1 Architecture

For this experiment, we use one encoder and separate decoders for the different languages. The architecture is the one shown in Figure 4. The primary task for this experiment is translation from English to German with the auxiliary task as translation from English to Vietnamese. The attention mechanism used is the Key-Value attention Vaswani et al. (2017). The setup has the bi-LSTM units having a hidden size of 512 units on encoder and both the decoders and the rest of the parameters are as described in part 5.1.

5.4.2 Training

The training is done with SGD optimizer with a batch size of 16 and a learning rate of 0.001, decayed to half on every epoch after 4 epochs. The decoder for primary task was trained 90% of the times, and the auxiliary task decoder 10% of the time and the loss calculated on each decoder is back-propagated only through that particular decoder and the common encoder and for a specific iteration the other decoder is entirely out of the picture.

5.4.3 Results

The multi-task model performs comparably with METEOR or BLEU-Score as the metric, as shown in table 5.

5.4.4 Observations

The primary observation is that by using Vietnamese as an auxiliary task we couldn't improve translations from English to German. The reason for this could be the fact that the Vietnamese corpus adds a large

Table 5: Single encoder and multiple decoders for different languages

Machine Translation Results for German		
Models	**METEOR 1.5**	**BLEU**
Without Vietnamese translation	46.91	26.35
With Vietnamese translation	46.54	25.82
Improvement	**-0.37**	**-0.53**

Table 6: Machine Translation Results for German : One to many system

Machine Translation examples
Source sentence 1: children are playing on a grassy field .
Target reference: kinder spielen auf einer wiese .
Model output: kinder spielen auf einer wiese .
Source sentence 2: the walking path along the beach is empty .
Target reference: der gehweg , der den strand entlangfhrt , ist leer.
Model output: die geht entlang am strand entlang .

number of words to the vocabulary which might be resulting in the embeddings learned being not so good due to the datasets being small.

Here are samples of translations of this model, one where the translation is perfect and other where it was bad, shown in table 6:

6 Conclusion and Future Work

Machine translation requires huge amounts of data in parallel corpus for good performance. Multi-task learning appears to provide richer feature representation and required regularization for the Machine Translation, when combined with some auxiliary tasks. This allows us to train a neural translation model on a small corpus, that also generalizes well. Both the auto-encoding tasks improved the translation performance when trained as auxiliary tasks. It suggests an inductive transfer of feature representations from language auto-encoding that can be useful for translation. However the POS tagging task didn't appear to provide such information. Our speculation is that this could be attributed to the nature of that task, where sparse intermediate feature representation are learned. Translation, on the other had, can perform better with the dense information provided by language auto-encoders.

We can think of a lot of improvements to these models: reversing the source sentence, optimizing for different attention mechanisms, using different learning rates for each of the encoders and decoders, sampling the decoder for better inference etc. to state a few. Encouraging results from this project suggest a wide variety of possible applications of multi-task learning across NLP tasks or multimodal tasks, provided the tasks are well related.

References

First conference on machine translation, 2016. URL `http://www.statmt.org/wmt16/`.

Abhinav, Sai Krishna Bollam, Sidhartha Satapathy, and Shivank Mishra. Code base for all tasks presented in this report. 2018. URL `https://gitlab-beta.engr.illinois.edu/ss46/MLinNLProject`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. Understanding and improving morphological learning in the neural machine translation decoder. In *IJCNLP*, 2017.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *ACL*, 2015.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CoRR*, abs/1705.07115, 2017. URL `http://arxiv.org/abs/1705.07115`.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL `https://doi.org/10.18653/v1/P17-4012`.

Mingsheng Long and Jianmin Wang. Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117, 2015. URL `http://arxiv.org/abs/1506.02117`.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogério Schmidt Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *CoRR*, abs/1611.05377, 2016. URL `http://arxiv.org/abs/1611.05377`.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114, 2015a. URL `http://arxiv.org/abs/1511.06114`.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015b.

I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, June 2016. doi: 10.1109/CVPR.2016.433.

Jan Niehues and Eunah Cho. Exploiting linguistic resources for neural machine translation using multi-task learning. *CoRR*, abs/1708.00993, 2017. URL `http://arxiv.org/abs/1708.00993`.

S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard. Learning what to share between loosely related tasks. *ArXiv e-prints*, May 2017.

Sebastian Ruder. An overview of multi-task learning in deep neural networks, May 2017. URL `http://ruder.io/multi-task/index.html`.

Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.