MIKE - MULTI-TASK IMPLICIT KNOWLEDGE EMBEDDINGS BY AUTOENCODING THROUGH A SHARED INPUT SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we introduce a method of learning Multi-task Implicit Knowledge Embeddings (MIKE) from a set of source (or "teacher") networks by autoencoding through a shared input space. MIKE uses an autoencoder to produce a reconstruction of a given input space optimized to induce the same activations in the source networks. This results in an encoder that takes inputs in the same format as the source networks and maps them to a latent semantic space which represents patterns in the data that are salient to the source networks. We present the results of our first experiments that use 11 segmentation tasks derived from the COCO dataset, which demonstrate the basic feasibility of MIKE.

1 INTRODUCTION

As representations, embeddings have proven useful in a wide range of agent architectures, facilitating information exchange between different architecture components such as world models and policy networks (Ha & Schmidhuber, 2018). Our goal in this work is to develop a scalable method of transferring the implicit knowledge from an arbitrarily large set of pretrained source/teacher networks to a single target/student network which will learn to produce embeddings of their knowledge. This is useful in scenarios where the goal is to absorb the representations of other networks without having to keep those networks in deployment.

Our method allows the source networks to be diverse both in terms of their topologies and the tasks they are trained on, with the constraint that all source networks must share the same input data format. The target network is assumed to be an autoencoder, with the encoder input and decoder output following the same data format as the source networks. It is also assumed that we do not have access to the datasets that were originally used to train the source networks. Instead, we have a "query dataset" which will be used to facilitate the knowledge transfer. Ideally, the query dataset will be representative of the domain to which we will apply our encoder. We refer to this as the query dataset because it is essentially used to "query" the latent knowledge of the source network so it can be transferred to the encoder. An ideal query dataset will result in the transfer of only knowledge that is salient to the tasks of the target domain.

Figure 1 summarizes how the encoder is trained and subsequently applied to the downstream target-domain tasks. The query data is passed through both the autoencoder and the source network(s). The reconstruction produced by the decoder is then also passed through the source networks. This produces activations in the source networks for both the original data and reconstructed data. The difference between these activations at various layers in the source networks is used to produce the loss which ultimately propagates back through earlier layers of the source networks and into the autoencoder, thus transferring the implicit knowledge of those layers. The information bottleneck in the autoencoder causes this knowledge to be represented as an embedding. This will ideally train the encoder to extract similar features as those of the source networks, thus distilling their implicit knowledge into the encoder.

Since the source networks will ideally have been pretrained on different tasks, we refer to this result as Multi-task Implicit Knowledge Embeddings (MIKE). The hope is that, by compressing the semantics of the source networks into a lower-dimensional representation, the implicit semantics of the representation will be easier for downstream components to decode, thus facilitating forward

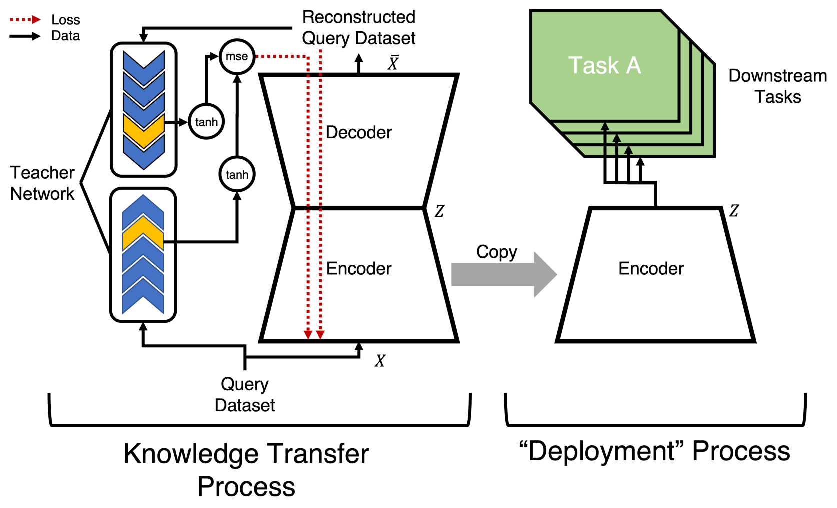

Figure 1: High-level summary showing how MIKE is trained and how the resulting encoder can be used to provide representations for downstream components/tasks. Note that the parameters of the source/teacher network are frozen during training of the autoencoder, and that the two teacher instances shown are in-fact the same network being evaluated on different inputs.

transfer to our target domain. Such embeddings could be applied to reinforcement learning agents, potentially serving as part of a world model (Ha & Schmidhuber, 2018), and may be particularly useful in scenarios where data is expensive to obtain, such as with physical robotic agents (Kalashnikov et al., 2018).

The secondary product of MIKE is a decoder, which can map the representations back to the input space in a way that preserves patterns which are salient to the source networks. We believe the decoder could potentially help human analysts intuitively understand the content of the representations. The data generated from the decoder might also find uses in automated processes within agents, such as memory replay mechanisms (Draelos et al., 2017) that are sometimes used to combat catastrophic forgetting in networks that learn continuously (Parisi et al., 2019).

The rest of this paper is organized as follows: Section 2 reviews Related Works pertinent to knowledge transfer and embedding learning in various contexts; Section 3 provides a formal description of the core Methods of MIKE, including high-level training topology (3.1), losses (3.2), and layerwise knowledge transfer algorithms (3.3). Section 4 details our specific experiments, describes how we produced our datasets (4.1) from Common Objects in Context (COCO) (Lin et al., 2014), and goes over the results (4.2). Section 5 contains a discussion of our findings including potential methods of addressing current limitations and some concluding remarks.

2 Related Works

Broadly speaking, MIKE applies learned similarity metrics (Larsen et al., 2016) to knowledge-transfer (Gou et al., 2021). The paper "Autoencoding beyond pixels using a learned similarity metric" replaced the generator for a Generative Adversarial Network (GAN) with an autoencoder, using features learned in the discriminator network to produce error for training the autoencoder. To generate error, the authors used the difference between the discriminators intermediate activations on original versus reconstructed data (referred to as "content error"). Instead of the intermediate activations of a discriminator, a later work (Hou et al., 2017) used the intermediate activations of a pretained ImageNet classifier, using the error to produce "feature perceptual loss". Within the context of this paper and using terminology we use here, the discriminator of Larsen et al. (2016) and the pretrained classifier of Hou et al. (2017) both constitute source networks, and both techniques use autoencoders to learn embeddings of the knowledge from their respective source networks. MIKE attempts to extend this general knowledge-transfer approach to all layers of multiple source networks.

It should be clarified that MIKE does not perform true distillation, since our target network (equivalently student network) does not explicitly learn to perform the task of the source network(s) (equivalently teacher network(s)). MIKE therefore resides in a middle ground between distillation and embedding learning, where the student network implicitly learns the knowledge of the teacher in the process of learning an embedding. Similar to many distillation methods, MIKE utilizes intermediate activations of the teacher networks in generating loss (Romero et al., 2014). However, this loss is generated between the teachers' own activations on the original and reconstructed data respectively, rather than between the teacher and student network activations for the same data.

We distinguish our use of the term *embedding* from its meaning in the context of Natural Language Processing (NLP). Our use is synonymous with "dimensionality reduction" whereas in NLP an embedding is usually a means of translating a categorical variable to a vector. When our embedding is learned from multiple teachers, the embedding may be referred to as a *joint embedding*. The term joint embedding often refers to multi-modal embeddings, such as those which combine an image and its textual description into a single unified representation (Weston et al., 2010; Socher et al., 2014). When we transfer implicit knowledge from multiple teachers into a single embedding, it is "joint" in the sense of being multi-task rather than multi-modal.

The term *latent semantic space* describes what our encoder provides a mapping to. The term *semantics* is typically associated with linguistics (e.g. syntax vs semantics), whereas we are using it to refer to the latent semantics of an image (i.e. image contents). Ultimately, semantics correspond to the salient regularities in a given scenario, which for an embodied agent would likely correspond to affordances (Min et al., 2016). This pragmatic relativism in semantics may be reflected in the differing manifold structures (Brahma et al., 2015) in teacher networks which were trained on different tasks. Therefore, a joint embedding of the implicit knowledge of the teacher networks constitutes a mapping to a latent semantic space which jointly reflects the semantics of all the teachers, attempting to merge their representations into a common space.

The resulting approach can also be interpreted as maximizing mutual information between the embedding and the activations of the teacher networks (Ahn et al., 2019). Granted, we are not directly maximizing mutual information, since the differing topologies between teacher and student make this infeasible. Instead this may be learned indirectly through autoencoding. By mapping from the input space to the embedding space and then back to the input space, irrelevant patterns are discarded while salient patterns (with respect to the teacher networks) are preserved. While not a direct method, MIKE may implicitly maximize mutual information between the embedding and the activations of the teacher networks, thus maximizing "mutual semantics" in a joint latent space.

3 METHODS

We alternatively considered two approaches to learning joint embeddings of source networks. However, both were discounted due to poor scaling characteristics and high resource usage. A direct approach would have been to use an autoencoder that takes the activations of the source networks as input, and attempts to reconstruct those activations. While preliminary experiments showed that this converges better, this approach poses a resource challenge since it requires the source networks to be retained in deployment. It also scales poorly since the number of parameters of the encoder and decoder grows with the combined size of the intermediate activation tensors of the source networks.

There is a hybrid approach where the encoder takes input X but the decoder attempts to predict the activations of the source networks. This implicitly distills the source networks into the encoder, but does not benefit from a decoder which is interpretable or interoperable. While this approach may produce a useful embedding, it also runs into training scalability problems due to the the growth in decoder parameters with the combined size of source network activation tensors. To avoid these disadvantages, we have instead chosen to autoencode through a shared input space using the training topology described in the following subsection.

3.1 TRAINING NETWORK TOPOLOGY

The left portion of Figure 1 shows the single-source-network base case of our training network topology (often called "student-teacher architecture" in distillation literature). The query dataset X is passed through the autoencoder to produce a reconstruction $\bar{X}$. Both X and $\bar{X}$ are then passed through the source network, resulting in two sets of activations over all the layers in the source network. A reconstruction $\bar{X}$ that preserves the patterns which are salient to the source network should produce activations that are similar to those of the original data X, with the difference in these corresponding activations serving as the basis for our loss function (3.2). MIKE generalizes this to multiple source networks in a straight-forward fashion by passing the query dataset through all source networks, and aggregating losses over the networks. The specific topologies of the source networks and autoencoder used in our experiments will be explained in Section 4.

The topologies of the source networks and autoencoder have one constraint; the input formats of all source networks and the input and output format of the autoencoder must be identical. For example, our experiments had an input space of images with dimensionality $3 \times 64 \times 64$. It is sometimes possible to accommodate source networks with different input space formats, but this generally requires a lot of work: multiple versions of the query dataset must be produced with the different formats; plus both the encoder and decoder require composite inputs that merge the different resolutions and formats of the data at downstream points in their topologies. In some cases it is also possible to perform network surgery on non-conforming source networks. For example, if we have a VGG-like source network pre-trained on ImageNet ($3 \times 224 \times 224$) and we want to map it to a $3 \times 64 \times 64$ input space, we can produce a subset of the network that truncates later layers which would cause downsampling below $1 \times 1 \times 1$. However, many types of data cannot be resized or reformatted, and the shared input space constraint remains one of the primary limitations of MIKE. As a last note on topology, while not technically a limitation, some extra implementation work is generally required to enable source networks to return an arbitrary set of their intermediate activations on their forward pass.

3.2 LOSS

The loss function (1) uses the difference between the activation tensors produced by the source networks for the original data, X, and reconstructed data, $\bar{X}$. Let N be the number of source networks, and L_n be the number of layers in a given source network, with $n \in \{0, \ldots, N-1\}$. With $l \in \{0, \ldots, L_n - 1\}$ indexing the layer, we assign a loss weight to a given layer denoted with w_l^n with the constraint that the loss weights sum to 1, i.e. $\sum_{n=0}^{N-1} \sum_{l=0}^{L_n-1} w_l^n = 1.0$.

We denote the activations of a given layer of a given source network applied to an input $x \in X$ with $\phi_l^n(x)$ (and $\phi_l^n(\bar{x})$ for $\bar{x} \in \bar{X}$ respectively). Our total loss $\mathcal{L}$ is then defined by

$$\mathcal{L} = \sum_{n=0}^{N-1} \sum_{l=0}^{L_n-1} w_l^n \left(\tanh\left(\phi_l^n(x)\right) - \tanh\left(\phi_l^n(\bar{x})\right)\right)^2 \tag{1}$$

though in training we average $\mathcal{L}$ over the batch dimension as is standard.

As seen in Figure 1 and Equation 1, we apply a hyperbolic tangent to the the activations of each layer in each network. This restricts their ranges between -1.0 and 1.0 and helps compensate for activations at different layers of the source network having different bounds, such as when comparing a sigmoidal layer to a ReLU layer. In most of our experiments, the weights for all the layers were set to be equal. However, it is possible to set the weights differently or even assign them dynamically.

3.3 LAYERWISE KNOWLEDGE TRANSFER ALGORITHMS

While MIKE usually assigns equal weight to all layers of all source networks, it currently also supports several knowledge-transfer-algorithms where we use dynamic loss-term weights during training. These other algorithms were prompted by convergence problems seen in early experiments attempting to directly absorb the knowledge from the last layer of a given source network. We found that pretraining the autoencoder on simple pixel-wise reconstruction error could help it converge more reliably. A generalization of this idea is the layerwise progressive algorithm for knowledge-transfer which reassigns the loss weight to progressively deeper layers of the source network at regular intervals during training. A more nuanced approach may be to change the layerwise loss weights adaptively, though this is beyond the current scope of this work. In the following section we describe our experiments, which use only two of the possible knowledge-transfer algorithms, namely “all-layer uniform” and “last-layer progressive”.

4 EXPERIMENTS

Our experiments thus far have aimed to test the basic feasibility of knowledge transfer using MIKE. The shared input space consists of matrices with dimensionality $3 \times 64 \times 64$, representing RGB color images (Figure 2). For our source networks, we pre-trained 11 three-class image segmenters on non-overlapping classes (i.e. different tasks), with all of these networks sharing the same topology. The segmenters and the autoencoder are both based on the EfficientNet (Tan & Le, 2019) block architecture, which was chosen for its compact model size and robust convergence. Our segmenter topology consists of 18 convolutional layers, including 16 EfficientNet blocks. It has a skip connection from the second layer which merges via concatenation with the output of the 15th layer, which we found to improve its performance at a minimal resource cost.

Within our autoencoder, the encoder consists of a 3×3 convolutional layer followed by eight EfficientNet blocks, with the final embedding being produced by 2-channel pointwise convolution, outputting an embedding matrix of size 128 and shape $2 \times 8 \times 8$. This encoder is unusual in that it is fully convolutional, producing a three-dimensional embedding rather than the typical flat embedding vector. This was found to converge to higher accuracies much more quickly, and require far fewer parameters than the equivalent encoder that used a dense layer. The decoder topology mirrors the encoder topology, though the parameters are not shared. We deliberately chose to keep our models small so the experiments could fit within the 16GB memory of our GPU. Using these particular networks, we were able to perform knowledge transfer from up to 11 source networks simultaneously, though we describe a potential method of overcoming this scalability limitation in Section 5. For reproducibility, we plan to open source our codebase at a future date.

4.1 DATASET

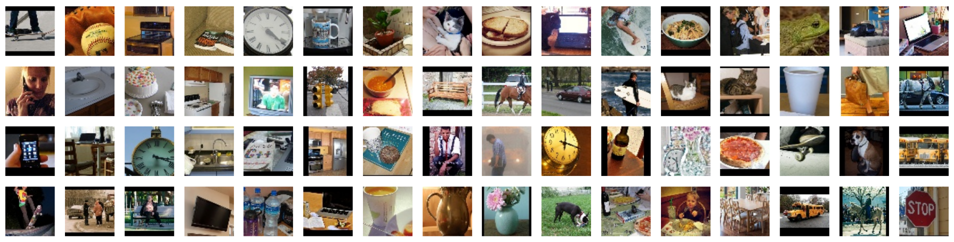

Figure 2: Various exemplar images from the 11 segmentation datasets that we derived from COCO.

The 11 datasets used to train each of our 11 segmenters were produced using the Common Objects in Context (COCO) API to obtain segmentation masks and bounding boxes for all the class instances in COCO. We cropped out these class instances and augmented them using shifting, zooming, and mirroring, before images were resized to 64×64 pixels (Figure 2 shows examples). While COCO contains 80 classes, we only used three from each of the 11 super-categories (ex. animals, furniture, vehicles, etc.). For example, from the “animals” category, we used the three classes with the most instances: cat, dog, and horse. These categories are: PHB: person, handbag, backpack; BTS: bench, traffic-light, stop-sign; SSS: sportsball, surfboard, skateboard; SOR: sink, oven, refrigerator; PCS: pizza, cake, sandwich; DCH: dog, cat, horse; CBB: cup, bottle, bowl; CDP: chair, dining-table, potted-plant; CTL: cellphone, tv, laptop; CTB: car, truck, bus; BCV: book, clock, vase.

In the interest of testing knowledge-transfer with as few confounders as possible, we used these 11 datasets for three different roles that would normally use distinct datasets. While we generally assume that source network training datasets are not available to facilitate knowledge transfer, our experiments relaxed this restriction by using the 11 datasets as query datasets for knowledge transfer. In each experiment, the query dataset is formed from a different subset of these 11 datasets, with the remaining datasets serving as downstream tasks which have a similar role to testing sets. However, for each autoencoder, we also show the results for downstream tasks which were included in the query dataset. This allows us to measure the amount of knowledge actually absorbed, while the non-query tasks measure how well the representations generalize. For example, in order to train an autoencoder via transfering the knowledge from the cat-dog-horse and car-truck-bus segmenters, the cat-dog-horse and car-truck-bus datasets would be used as the query dataset. To ensure a fair comparison, the baseline autoencoder with simple pixel-wise error would also be trained on cat-dog-horse and car-truck-bus.

One consequence of pulling multiple datasets from COCO is the potential for roughly the same image to appear in more than one of the 11 datasets, but with different labels. When we are attempting knowledge-transfer from more than one source network, these conflicting samples may end up together in the query dataset. To mitigate this, our particular experiments modified the loss calculation so that losses from each of the source networks were only calculated based on the subset of the query dataset which overlaps with the source networks training set. In other words, we only want the loss to care about the way each source network reacts to its own training data, and not how it reacts to the training data of the other source networks. This prevents different labelings of the same image from producing conflicting gradients within a given source network.

4.2 Results

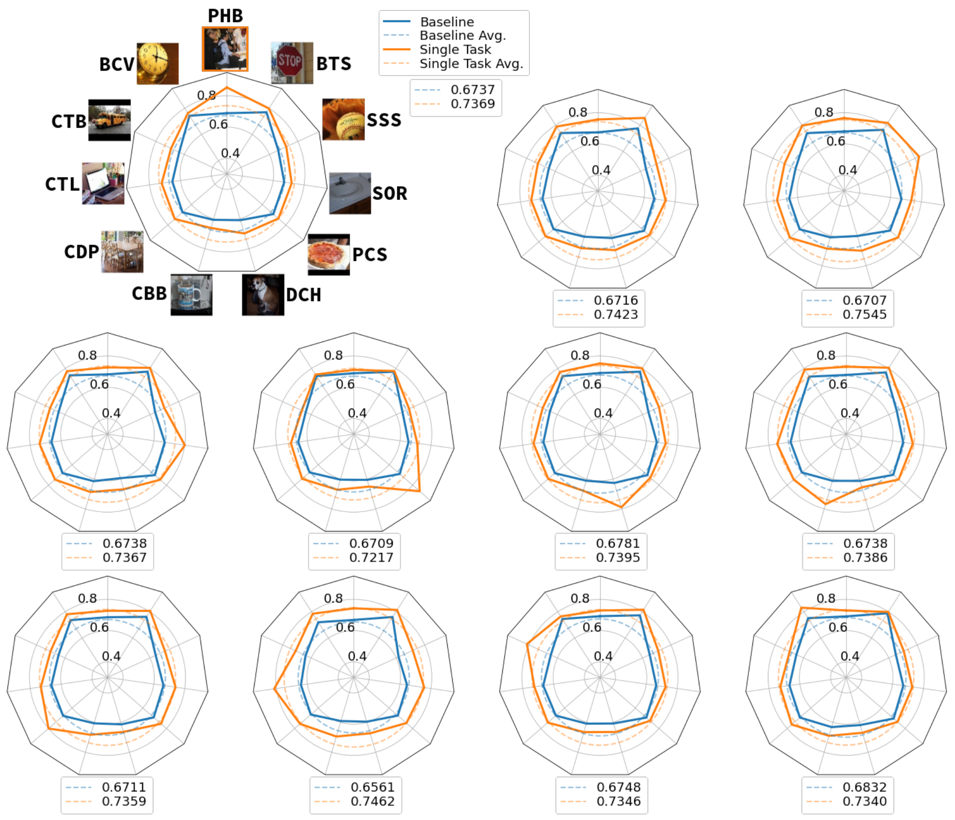

Figure 3: Radar charts comparing the downstream performance across the 11 coco segmentation tasks. Each of the 11 radar charts is for a single encoder that absorbed the knowledge from only one of the segmenter source networks, with the left-to-right/top-to-bottom order corresponding to the clockwise ordering of the upper left radar chart. These encoders were all trained using the all-layers uniform “knowledge-transfer algorithm”. They all outperform their corresponding baseline autoencoder on every downstream task, indicating that generally useful segmentation features were learned. The average task performance (measured in IOU) for the baseline and MIKE encoders are shown under each radar chart.

In our first experiment, we used MIKE to train 11 different autoencoders, each absorbing the knowledge from just one of the 11 pre-trained segmenter networks. We then used each of these encoders and their embeddings to train 11 downstream networks to perform the same 11 segmentation tasks. Normally, a query dataset would be distinct from the training datasets of the source networks (since said training sets may not be available) and would be chosen based on the intended target domain that the encoder will be applied to. However, in these experiments we used the same segmenter/source network training sets to simplify performance comparison. Each autoencoder was converted into a segmenter by freezing the encoder and reinterpreting/retraining the decoder to produce a three-channel segmentation mask instead of an RBG image, which was convenient since it required no changes to decoder topology. Using the decoder architecture as the downstream network also removed any uncertainty regarding the capacity of the downstream network to adequately decode/map the latent embeddings to other spaces.

The results of these 11 experiments are shown in Figure 3. We measured accuracy of the downstream task as the Intersection Over Union (IOU) for the downstream segmenter (i.e. repurposed decoder) divided by the IOU of the original segmenter that we trained on that same dataset. Using the original segmenter accuracies to normalize the downstream segmenter accuracies makes comparison between tasks easier and provides a more direct way of measuring the amount of knowledge transferred to the autoencoder in the case where the downstream task and source network task were the same.

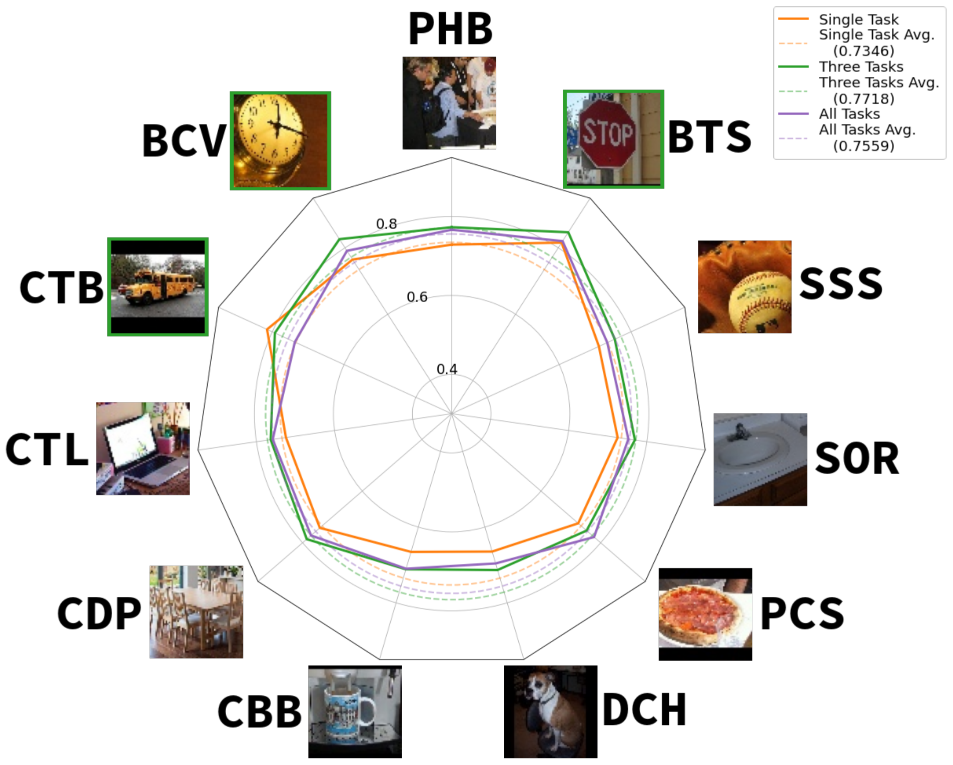

Figure 4: Radar chart comparing the downstream performances of three autoencoders trained on different numbers of source networks (i.e. tasks). The single task encoder was trained against the car-truck-bus source network. The three task encoder was trained against the three tasks whose exemplar images have a green border. The final encoder was trained against all 11 source networks. The results indicate that knowledge from multiple source networks can be absorbed simultaneously.

Our first set of experiments indicated that MIKE could transfer knowledge from a single source network to an autoencoder. Our second and third experiments have begun to test the ability of MIKE to transfer knowledge from multiple source networks into one autoencoder. Figure 4 compares our results for a three-network transfer and an 11-network transfer to a single-network transfer. The best average performance on the 11 downstream tasks was seen in the three-network transfer.

Interestingly, the 11-network transfer performed slightly worse, though still performed better than any of the single-network transfers shown in Figure 3.

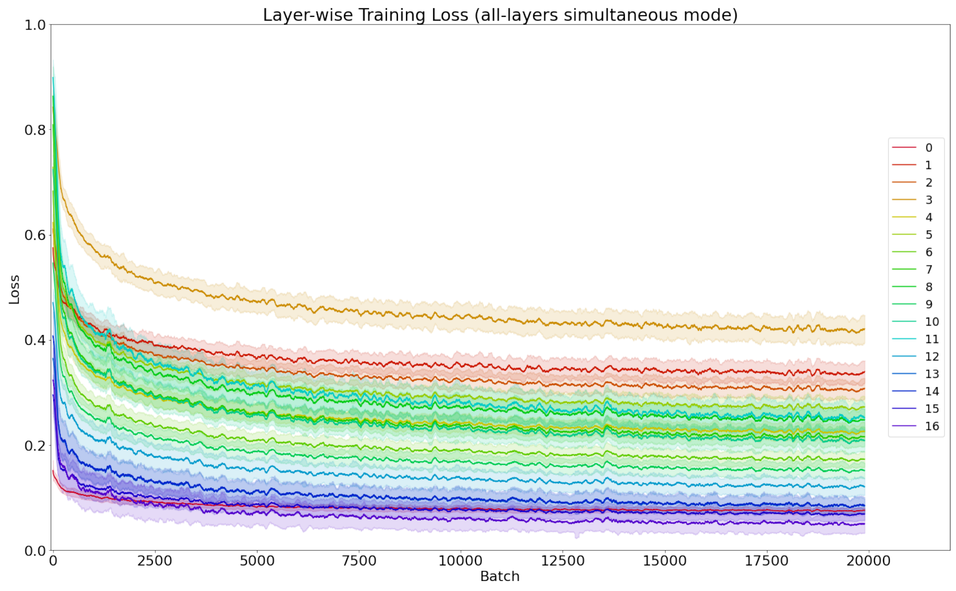

Figure 5: Layerwise training loss during knowledge transfer from the bench-trafficlight-stopsign source network. Loss weights were equal for all source network layers throughout training.

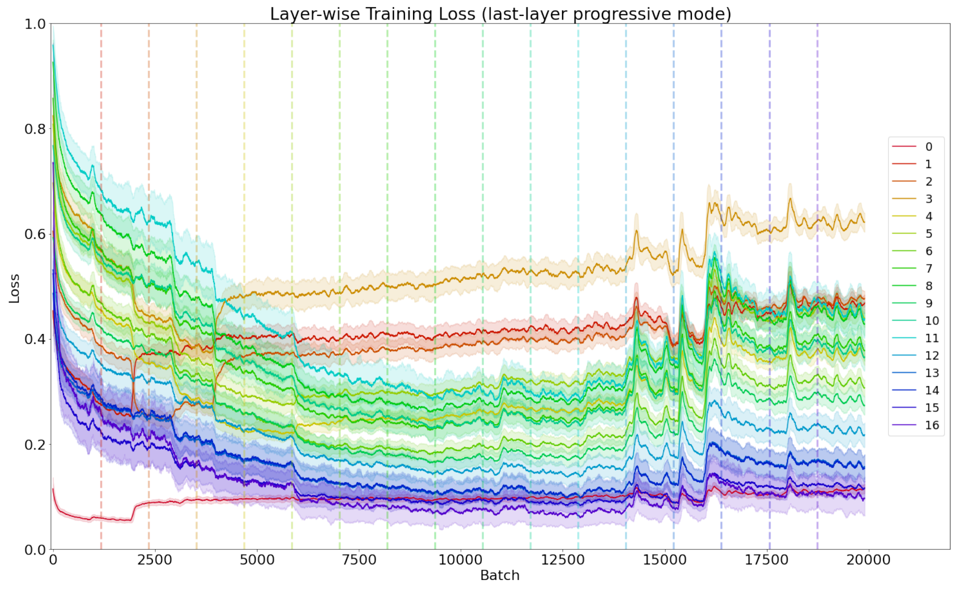

Figure 6: Layerwise training loss during knowledge transfer from the bench-trafficlight-stopsign source network. Loss weight was placed on only one layer at a time, moving sequentially through the network at regular intervals.

Our fourth experiment compared convergence for the all-layers uniform loss weighting (Figure 5) to that of the last-layer-progressive knowledge-transfer algorithm (Figure 6). The goal of the last-layer-progressive experiment was to absorb the information from the last layer of the network, which should also absorb much of the information from earlier layers by proxy. This was motivated by our expectation that the mutual information between a given set of activations and the output space should increase for activations that are closer to the output of the network. Unfortunately, convergence became very unstable as loss weights were adjusted for deeper layers.

5 Discussion and Conclusion

Our initial experiments have demonstrated the basic feasibility of MIKE, though considerable follow-on is required before practical deployment. Even for the single source-network case, there is evidence of gradient instability, particularly from the deeper layers of the source network. We hypothesize that this instability is exacerbated by the dimensionality-reduction induced information bottleneck at the center of the autoencoder. A simple way to test this hypothesis is to systematically vary the size of the embedding space. If this improves gradient stability, then other approaches to information bottlenecks may be viable. For example, perhaps an overcomplete autoencoder with an information bottleneck produced by sparsity regularization will impact the gradient differently. A potential consequence of poor gradient stability combined with a high-capacity decoder is that the decoder (rather than the encoder) may potentially absorb knowledge from the source networks. If so, then our experiments may be significantly underestimating the amount of knowledge actually absorbed by the autoencoder as a whole. A mirrored autoencoder may prevent asymmetry in knowledge absorption.

Another avenue that deserves more exploration is the influence of the relative capacities of the encoder and decoder on the resulting encoding scheme of the embeddings. Some of our preliminary experiments have shown that downstream networks with similar topologies to the decoder are more effective at decoding the embeddings, as evidenced by their higher performance on the downstream tasks. This suggests that a complex decoder function may cause the encoder to learn embeddings that are similarly complex to decode. We plan to perform experiments which vary the capacity and topology of the decoder, with the general hypothesis that the optimal decoder should mirror the structure of downstream network(s) that will be consuming the embeddings in the target application.

Finally, infeasible training times are quickly encountered if all source networks cannot fit in memory. In such a case, sequential knowledge transfer may be used, however, like other sequential learning tasks, catastrophic forgetting will likely become a challenge. Methods such as elastic weight consolidation (Kirkpatrick et al., 2017) could mitigate this, which should be simple since the task transitions are known a priori. As the elasticity of the weights decreases with each task, structural plasticity methods may increase the capacity of the autoencoder. While this could potentially grow the embedding space to the point of being overcomplete, an information bottleneck would still be supplied by the elasticity parameters of the weights.

References

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D. Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer, 2019.

Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. Why deep learning works: A manifold disentanglement perspective. *IEEE transactions on neural networks and learning systems*, 27 (10):1997–2008, 2015.

Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Jonathan A Cox, Craig M Vineyard, Kristofor D Carlson, William M Severa, Conrad D James, and James B Aimone. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 526–533. IEEE, 2017.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

David Ha and J Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1133–1141. IEEE, 2017.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Anders Boesen Lindbo Larsen, Sren Kaae Snderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *Proceedings of the 33rd International conference on machine learning*, 48:1558–1566, 2016.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Huaqing Min, Ronghua Luo, Jinhui Zhu, Sheng Bi, et al. Affordance research in developmental robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 8(4):237–255, 2016.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.