The Dissertation Committee for Christopher Edwin Crabtree
certifies that this is the approved version of the following dissertation:

Optimizing Visual Grounding of Latent Representations of Speech from Distant Language Groups

Committee:

David Harwath, Supervisor

Greg Durrett

Optimizing Visual Grounding of Latent Representations of Speech from Distant Language Groups

by

Christopher Edwin Crabtree, B.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2021

Dedicated to my wife Shirley.

Acknowledgments

I wish to thank the multitudes of people who helped me. Time would fail me to tell of ...

Optimizing Visual Grounding of Latent Representations of Speech from Distant Language Groups

Publication No. _____

Christopher Edwin Crabtree, Ph.D.
The University of Texas at Austin, 2021

Supervisor: David Harwath

This document has the form of a "fake" doctoral dissertation in order to provide an example of such, but it is actually a copy of Miguel Lerma's documentation for the Mathematics Department Computer Seminar of 25 March 1998 updated in July 2001 and following by Craig McCluskey to meet the March 2001 requirements of the Graduate School.

This document and its source file show to write a Doctoral Dissertation using LaTeX and the utdiss2 package.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent years have seen an increasing research interest into using multi-modal grounding techniques to bolster classic natural language processing (NLP) and automated speech recognition (ASR) tasks (CITEME). Among these have been efforts to use information from visual modalities such as images to improve the performance of neural machine translation (MT) and ASR models (CITEME). These research efforts are often motivated by the intuition that humans most often learn language through processes that involve resolving relations and co-occurences between input from a source modality (most typically spoken language), with that of input from sensory obtained from their environment.

Therefore it is a reasonable hypothesis to assert that the addition of co-occurring contextual information from modalities such as vision would be likely to aid in many natural language understanding tasks. However, (Caglayan 2019 "Probing...") noted that research efforts, starting with the multimodal machine translation (MMT) shared task of the First Conference on Machine Translation have not yet shown that that the addition of co-occurring visual information substantially improves MT performance. (Caglayan) did show however, that when linguistic information is scarce or noisy models using visual information are better able to recover from

the missing information during translation. Additionally, (Huang et al "Unsupervised MMT...") showed that in unsupervised MT settings they were able to incorporate the visual modality to consistently improve translation without aligned corpi for training. Given the linear nature of modern neural architectures, these results would seem to indicate that latent visual representations are able to, in some fashion, improve alignments of the embedded representations of sentences in different languages.

Similarly in the ASR domain, (Srinivasan, "Fine-grained grounding...") found that while visual information did not necessarily improve performance of speech recognition on clean datasets, when degrading the audio input using word masking their novel neural architecture was better able to recover the original word when leveraging features from a visual modality.

These empirical observations seem to suggest that, contrary to intuition, additional information (in the form of co-occurring modalities carrying mutual semantic information) does not substantially effect neural models' ability to recognize and utilize the patterns present in the primary source modality.

Parallel lines of research, though, into multilingual machine translation have shown that the addition of new language pairs *does* tend to improve transnational performance of all models. This phenomena has been termed transnational knowledge transfer (Dabre, "A comprehensive Survey ..."). This was first shown by ... (Google zero shot paper) and also .... These results align, very generally speaking, with the intuition that richer information sources should enable pattern recognition algorithms, such as neural networks, to more quickly lock on the tiny variations in the input that are semantically relevant.

This line of reasoning is further bolstered by results by (Harwath interlingua) which showed that the addition of visual context (in the form of pre-trained image features) approximately doubled the recall scores of cross-lingual audio retrieval between Hindi and English image descriptions. (Ohishi et al.) also showed that retrieval scores between audio-visual and visual-audio modalities improved for all pairs when augmenting (Harwaths) dataset with Japanese descriptions.

As of yet it remains unclear what extent visual grounding might aid in ST and MT, though, or what might be the best way to incorporate visual information into these tasks.

In this work we investigate task of image and multi-lingual utterance retrieval. Concretely, in image retrieval the task is to take a spoken utterance that describes an image and retrieve the corresponding image from a pool of candidate images (in our case the pool is the entire annotated dataset). The dataset used contains three descriptions of each image, each description in one of three distinct languages: English, Hindi, or Japanese. Likewise in multi-lingual utterance retrieval, the task is to retrive an utterance in a target language that describes the same image as an utterance in a given source language utterance. This task does not use the mutual image at inference time, but does not prohibit the use of the shared image during training.

It is worth emphasizing that in this task and in our dataset there is no assumption of lexical or syntactic alignment between language utterances. That is, speakers may choose to describe the same image in completely different manners, using words and expressions that do not directly align with what was spoken in another

language about the same image. The lack of syntactic alignment means that lexical and syntactic co-occurence information will be largely inconsistent or even absent. Nevertheless, since every utterance share the same grounded reference there should intuitively be consistent mutual semantic information shared amongst each training image and utterance triplet.

The task of retrieval was chosen because it can both serve as a proxy to measure latent representation alignment (since similarity measures are inherently used in the retrieval step), and also has more immediate practical applications, specifically in utterance to image retrieval. Using spoken utterances for data retrieval typically requires a large amount of hand engineering of many sub-components. These include, but are not limited to: acoustic, pronunciation, and language models. Predictive distributions between each sub-component are often combined using a large finite state transducer. The task of creating a new ASR system is often a laborious task that can be both time and data intensive. Furthermore, once an ASR system is built, at least using current technology, the highly tuned sub-components cannot be repurposed into submodules of ASR systems for other languages. Thus an entirely new system must be built for each new language.

This work explores instead the extent to which latent feature representations from distant languages can be embedded into the same semantic vector space. End-to-end (E2E) neural network encoders are used to produce these representations, thereby eliminating much of the sub-component specific implementation overhead needed to build classic ASR models for retrieval.

There are still many aspects of E2E ASR models, though, that are not well un-

derstood, especially in multi-lingual settings. In particular, it the exact translational mechanism in modern neural models is difficult pinpoint so it is unclear whether mutually aligned semantic embedding spaces are a As (variations bounds on MI) has noted, even the general task of optimizing MI has shown to have deteriorating and even degrading returns in downstream tasks.

This lack of clear understanding motivates the experiments presented in this work. In particular, this work attempts to address three questions:

1. What type of learning objective (i.e. loss function) results in the best alignment of image and multi-lingual utterances (as measured by average retrieval scores)?

2. Since contrastive learning objectives rely on pairs of negative examples, aligning multiple modalities in the same representation space suffers a exponential increase in loss terms with the pairing each new modality. To what extent can this be mitigated?

3. Similarly, it is reasonable to assume that at least some of the task of modelling linguistic information might be shared in internal model parameters (e.g. recognizing phonemes shared between languages), which is one explanation for results demonstrated in previous work on translational knowledge transfer. Can this phenomena be leveraged to reduce the model size/memory requirements of aligning a new language to the shared representation space?

Chapter 2

Background

This section will review the primary INSERT LATERthat will be compared empirically through the experiments in INSERT LATERput in refs

2.1 Task Description

All experiments in this work are designed to estimate predictive performance on retrieval tasks. This section formalizes the retrieval task and describes it in detail for the sake of clarity. We also define terms that well be used for the rest of the paper.

We are given a Dataset, $\mathcal{D}$, containing a set of K distinct *Views*. We will refer to an arbitrary view as $\mathcal{V}_j$ where $j \in [1, K]$ indexes each view. Each view, $\mathcal{V}_j$, contains N samples and each sample, $\nu_{ij} \in \mathcal{V}_j, i \in [1, N], j \in [1, K]$ has a corresponding entry in each of the other $K - 1$ views which share some sort of semantic relatedness. Furthermore, each V_j has it's own representation scheme for it's constituent samples which making a conventional tensor representation with fixed dimensions for all samples in D inexact. This gives the generalized dataset for our task the following definition:

$$\mathcal{D} = \{(\nu_{i1}, \nu_{i2}, \cdots, \nu_{iK}) : \nu_{i1} \in \mathcal{V}_1, \nu_{i2} \in \mathcal{V}_2, \cdots, \nu_{iK} \in \mathcal{V}_K, \quad i \in [1, N]\}$$

To reduce ambiguity, we will refer to an arbitrary K-tuple in $\mathcal{D}$ as a *datapoint* and

an arbitrary member of a K-tuple/datapoint, ν_{ij}, as a *viewpoint*. We will be sure to clarify in cases where this definition of viewpoint allows for ambiguity between the colloquial use of viewpoint by instead using 'point of view' for the colloquial sense as needed.

To be more concrete for a moment, in the dataset used for this work we have $K = 4$ views in our dataset which are: 1) the set of images, 2) the English descriptions, 3) the Hindi descriptions, and 4) the Japanese descriptions (although we do not assume that particular ordering). For each image (viewpoint) in the image view, there is a corresponding spoken utterance (viewpoint) describing that image in all three languages. We therefore assume some type of reliable mutual semantic information between each viewpoint in all datapoints.

As an aside, *mutual information* (MI) has a well-known and well-defined mathematical form, but we will be using the term in the idiomatic sense. Recent work by (CITEME) has shown that many popular lower-bound maximization schemes for estimating MI break down in practice and do not correlate with expected performance gains. Furthermore, MI is only well-studied in the two variable (view) case. The generalization of MI to three or more variables is referred to as *interaction information* (II) and has certain properties (such as permitting negative values of II) that make it difficult to interpret.

Also note that, on occasion, we further categorize our views based on *modality*. To be clear, this dataset contains two modalities: vision and audio; one view is in the vision modality (images) and three views are in the audio modality (English, Japanese, and Hindi utterances). We therefore commonly differentiate retrieval re-

sults based on modality pairing, retrieval results from vision-audio pairs being referred to as 'image retrieval' and audio-audio retrieval results referred to as cross-lingual.

Keeping these points in mind, we still wish to encode the datapoints from each view into a shared vector space. Our task, then, in words is: given an input (indexed by i), ν_{ij}, from the j^{th} view, V_j, and a set of datapoints from a target view V_t, we would like like to retrieve ν_{it} from V_t. In order to do this we use an *encoder* function f_θ to produce a numeric representation of a viewpoint.

f_θ is parameterized by an arbitrary number of parameters denoted by θ. It takes a viewpoint as input, and produces an output representation. In this work all encoders take the form of neural networks, and in particular CNNs reference descrip in background. We also allow the encoder to take as input information indicating the input view. That is, if we let j denote the input view, we can denote the encoder and it's input/arguments as $f_\theta(\nu_{ij}, j)$. This notation effectively generalizes over many potential implementations of f_θ. f_theta might be implemented such that all views have a distinct set of model parameters, or some parameters might be shared across a subset views.

The retrieval is then performed by comparing the input, ν_{ij}, with all other $\nu_{ik} \in V_k$ and returning the ν_{ik} with the highest *similarity score*. This similarity score is measured by a chosen *similarity function*, S. S follows the typical mathematical definition of a distance measure (i.e. it takes two arguments, it's always non-negative, and it is symmetric in it's arguments). For retrieval tasks this similarity function typically takes one of the following forms: the dot-product, the euclidean distance, or the cosine similarity.

With all terms defined, we can write our most general leaning objective as:

$$\max_{\theta}\Big(\sum_{(\nu_i 1,\cdots,\nu_i k)\in D, i\in[1,N]}\ \sum_{j\in[1,K]}\max_{t\in[1,K],t\neq j} S(f_\theta(\nu_{i,j},\, j), f_\theta(\nu_{i,t},\, t))\Big) \tag{2.1}$$

Note, again, that f_θ need not be a single neural network encoder.

This optimization objective has no known exact solution find support or use other language to describe difficulty and must be approximated. Recently, this type of optimization has been routinely approximated using a differentiable loss function and a gradient-descent based optimization algorithm. There are a number of loss-function in common use for this type of maximization objective, many of which are motivated by maximizing the estimate of the aforementioned MI estimate. Which one of these loss functions produces optimal results for our particular setting is the subject of the following experiments in this chapter.

2.2 Loss Functions

There are numerous loss functions that are designed to encourage 'similarity' among pairs or groups of datapoints. We explore a subset of these in this work an describe the important aspects of each below.

2.2.1 Triplet

Triplet loss has perhaps the longest history of use in this task. Harwath et al. [4] first introduced the task of spoken image retrieval and used a maximum margin objective between each final unpooled image embedding and final sequence embedding of paired images and utterances. This obective eventually became formulated a the

the triplet loss.

Generally, triplet losses take an *anchor* embedding/vector A, *truthy* embedding T (which is positively associated with the anchor), and a *falsey* embedding F, also known as an imposter, that is not associated with the anchor. Then, given a chosen similarity function, the triplet loss can be calculated as:

$$Triplet(A, T, F) = \max(0, S(A, F) - S(A, T) + M) \tag{2.2}$$

Where M is some margin to be chosen a hyperparameter (often just 1). This loss effectively pushes the A and T embeddings toward each other and the A and F embeddings away from each other. The margin, then, influences how stronly model is encouraged to push or pull the associted embeddings.

This was formalized in [7] for spoken image retrieval by using two triplet losses: one defined with image representaions as anchors, and one with spoken captions as anchors. It can be written as:

$$T(I_j, C_j, I_j^{imp}, C_j^{imp}) = \sum_{j=1}^{N} \Big(\max(0, S(I_j, C_j^{imp}) - S(I_j, C_j) + 1) \tag{2.3}$$

$$+ \max(0, S(I_j^{imp}, C_j) - S(I_j, C_j) + 1) \Big) \tag{2.4}$$

Where S is a chosen similarity function, I_J and C_j are the j^{th} image/caption which are treated as , and I_j^{imp} and C_j^{imp} are the j^{th} *imposter* samples of an image and caption not associted with the correct image/caption pair. In this formulation, the first max term is effectively the triplet loss when I_j is viewed as the anchor, C_j is viewed as T, and C_j^{imp} is viewed as F. The second max term then uses C_j as the

anchor and the images in likewise fashion. The imposters are sampled from a uniform distribution from within the minibatch.

This form of triplet loss is highly dependent on the imposter samples chosen, since in the final stages of optimization most negative samples will alredy be far from the anchor. This motivates hard and semi-hard negative sampling [12] which, generally speaking, seek to sample imposter/negative samples that are very close to the anchor (and there for 'hard' to distiguish from the positives). This general approach was adapted to the spoken image retriaval task by Harwath et al. [6] which showed that the use of semi-hard negatives increased retriaval performance.

The accompanying loss function, was formulated more generally in [5] as:

$$\mathcal{L}(\theta) = T(I_j, C_j, I_j^{imp}, C_j^{imp}) + T((I_j, C_j, \tilde{I}_j^{imp}, \tilde{C}_j^{imp}) \tag{2.5}$$

Where T is the same as in Equation 2.3. The important difference between the terms is that $\tilde{I}_j^{imp}$ and $\tilde{C}_j^{imp}$ are chosen to be the image and caption that are most similar to their respective anchors in the batch (as measured by the similarity function S), instead of the uniform distribution as I_j^{imp} and C_j^{imp}

2.2.2 Masked Margin Softmax (MMS)

The Masked Margin Softmax (MMS) loss was introduced by Ilharco et al. [11] and uses many more negative examples than triplet loss. MSS loss effectively computes the categorical cross-entropy loss between an anchor embedding and K other embeddings, only one of which is the positive corresponding embedding. Letting B be the batch size and X and Y be the final representations of B samples of two arbitrary

informations sources. We use information souce to refer to one of the followinge: an image, or a speech utterance from a language in the set of $\mathcal{D}_{lang}$ languages in the dataset. MMS can then be written for a single minibatch as:

$$MMS(X,Y) = \frac{1}{B}\sum_{j=1}^{B} \frac{e^{S(X_j,Y_j)-M}}{e^{S(X_j,Y_j)-M} + \sum_{k=1}^{B} \mathbb{I}[k! = j \wedge k \notin \mathcal{Z}] e^{S(X_j,Y_k)}} \tag{2.6}$$

Where $\mathcal{Z}$ is a set of predefined indices corresponding to examples to be *masked* (hence the name). In our dataset there are no examples of this sort so $\mathcal{Z}$ is always empty.

Notice that $MMS(X,Y)$ is not symmetric. Specifically, each representation X_j is contrasted with all other Y_k in the batch, by Y_j is not given the same treatment. The full MMS loss for a batch completes the symmetry and is defined as:

$$\mathcal{L}_{MMS}(\theta) = MMS(X,Y) + MMS(Y,X) \tag{2.7}$$

The margin M is an important quantity and can be a constant value, updated according to a schedule, or adaptively updated according to some function of the batch. When $M = 0$ this loss has the form of what is often referred to in the literature as InfoNCE loss [18]. Several works use this terminology [8, 10], but there seems to be a lack of consensus over this issue [21]. Still, we hesitantly follow the current conventions and whenever an experiment uses a margin of 0, we will refer to it as InfoNCE.

These methods are explored empirically in Section 3.1.5.

2.2.3 Hypersphere Loss

The hypersphere loss was introduced by Wang [23] and attempts to reframe conventional contrastive loss through the notions of *alignment* and *uniformity* of la-

tent representaions on the hypersphere. They provide empirical evidence that directy optimizing their measures of alignment and uniformity can outperform contrastive losses. Given a batch of size B with X and Y defined in the same manner as in Section 2.2.2, the *alignment* sub-loss is defined as:

$$HSphere_{align}(X, Y) = \frac{1}{B} \sum_{i=1}^{B} \|X_i - Y_i\|_2^{\alpha}$$

where α is a hyperparameter to be tuned.

The *uniformity* sub-loss is calculated as:

$$HSphere_{uniformity}(X, Y) = \frac{1}{B(B+1)/2} \sum_{i=1}^{B} \sum_{j \geq i}^{B} e^{-t\|X_i - Y_j\|_2^2}$$

with t being another hyperparameter.

In the original work the showed that values of α and t around 1 tended to work well, but t often needed to be lower than *alpha*.

With those defined the full hyperspheric loss is defined as:

$$HSphere_{full}(X, Y) = \lambda_a * HSphere_{align}(X, Y) + \lambda_u * HSphere_{uniformity}(X, Y) \tag{2.8}$$

With λ_a and λ_u being additional weiting parameters. In the original work, an additional baseline contrastive loss was also added to Equation 2.8 with an accompanying weighting hyperparameter, but they found that it was not necessary for optimal performance so we have omitted it in our experiments for simplicity.

2.3 Multiview Contrasting Frameworks

Work by Tian et al. [21], provided a framework for using contrastive loss functions with several *views*. They define a view as a certain sensory input during an arbitrary event. In this way, there may be multiple views all describing the same underlying (or latent) event. A view, then, can have a more abstract meaning in which some amount of information contained in the view is shared among other views.

Their work, then attempts to devise appropriate training schemes to maximize the latent information shared by all views. The two components of their work relevant to ours are the *full-graph* and *anchor* training schemes.

Both schemes start with a symmetric loss function, $\mathcal{L}(\mathcal{V}, \mathcal{V}$, defined over two views, V_1 and V_2, of size B. Each samples in V_1 are assumed to have a corresponding positive sample in V_2 that carries the same mutual information (and vice-versa for the samples in V_2). Note that V_1 and V_2 are from a minibatch of the full dataset, but to reduce notational clutter we will only define a single batch update. $\mathcal{L}$ is also assumed to be designed such that it encourages alignment between the positive examples in two views and some notion of distance between the examples that do not carry mutual information. We will assume as well that the positive examples share the same index in the batch. Notice that Equations 2.8, 2.5, and 2.7 all meet the above assumptions.

Finally, the full-graph and anchor frameworks attempt to maximize the mutual information contained in a set of K views $\mathcal{V}$.

With the above assumptions and notation, the full graph framework computes

the following loss for each batch:

$$\mathcal{L}_{full_g}(\theta) = \frac{1}{B(B+1)/2} \sum_{i=1}^{B} \sum_{j \geq k}^{B} \mathcal{L}_s ym(V_i, V_j)$$

The anchor framework can be defined intuitively as aligning all views to one 'anchor' view in a hub-and-spoke manner. It is calculated as:

$$\mathcal{L}_{anchor}(\theta, i) = \frac{1}{B-1} \sum_{j=1}^{B} \mathbb{I}[j \neq i] \mathcal{L}_s ym(V_i, V_j)$$

Where i denotes the index of the anchor view.

2.4 Neural Architectures

We use convolutional neural networks (CNN) as our base architecture. The focus of this work is on the effect of loss functions and training regime in a multilingual image retrieval and as such does not attempt to optimize architecture. We therefore use two off the shelf encoders for our images and speech utterances.

2.4.1 Image Model

The image encoder is based on the ResNet50 model proposed by He et al. [9]. It is pretrained on ImageNet classification, but the final layer has been altered slightly. In the same manner as Harwath et al. [5], we remove the final softmax and fully connected layer and instead perform a 1x1 convolution to obtain a desired size of the final imagage representations. The output representation size for each image is a 7x7x1024 tensor.

2.4.2 Audio Model

We will first describe the pre-processing steps taken on the audio and then briefly describe the architecture.

2.4.2.1 Audio Pre-Processing

The same pre-processing is used for all experiments. Log-Mel filter bank spectrogram features are calculated from raw audio with 40 frequency bins. We use Hamming-windowed frames with a 15 ms width and shift of 10 ms. During batching, all utterances are padded up to the longest utterance in the batch.

2.4.2.2 architecture

The audio/speech model we use the same as [5]. It is a 17 layer CNN with the first being a 1D convolution of 128 filters of size 1x40x1 across the time dimension. The following 16 layers divided into 4 residual *speech* blocks. Each speech block starts with an initial 1D convolution layer with kernel size of 1x9 a stride of two that effectively downsamples the input followed by a batch norm layer. This followed by three successive 1D convolution (kernel size 1x9, stride 1) and batch norm layers. No padding is used. This produces 1024 dimensional embeddings of various length depending on the batch.

Chapter 3

Learining Objectives

This chapter explores the extent to which the overall learning objective effects retrieval performance, indicating better alignment with respect to a similarity metric. We note that the terms 'learning objective' and 'loss function' are used in this chapter interchangeably.

3.1 Experimental Design

3.1.1 Data

The experiments in this chapter (and all proceeding chapters) use a dataset consisting of 100,000 images from the Places 205 dataset (CITEME) that have additional English (CITEME), Hindi (CITEME), and Japanese (CITEME) spoken descriptions corresponding to each image. It is important to note that the descriptions from each language are not translations of each other as each speaker is only directed to describe the image and has no knowledge of what was said previously. This means that there is no assumption of alignment in the ordering of objects described or the way in the manner in which the speaker chooses to describe the image. The statistics for the descriptions can be found in (CITEME)see if all stats are in one place

We use a training set of 99,000 datapoints and use the remaining 1,000 data-

points for testing. In cases such as INSERT LATER in which hyperparameter tuning is performed, a random subset of 1,000 datapoints are extracted from the training set and used for evaluation.

3.1.2 Models

The experiments in this chapter all use the same basic implementation for the full encoder function f_θ. All viewpoints of the image view are encoded using an image-specific encoder. Additionally, each language view has it's own dedicated speech encoder, meaning there are no shared layers/parameters across languages. However, the architecture of each language's encoder is identical.

All models used in this work are convolutional neural networks (CNNs) at their core. The details of each CNN architecture are given below.

3.1.2.1 Image Encoder

The image encoder we use is a resnet CNN published by (He et al) and is initialized using the weights of a model that has been pre-trained on imagenet (CITEME) object classification. The image encoder has INSERT LATERarchitecture specs here. For our dataset the representation of each image output by the image encoder has fixed dimensions and is in $\mathbb{R}^{7\times7\times1024}$. Where 1024 is out embedding size of each super pixel.

3.1.2.2 Audio/Speech Encoder

The audio encoder architecture is the same across all languages. It is a resnet CNN introduced by (Harwath resnet paper). The architecture is similar to the image encoder, but with slight differences in the dimensions in each layer. These differences are a natural result of the inherent differences in the visual and auditory modalities. The details of the architectural differences are given in figure make and reference figure. All audio encoders are randomly initialized.

Unlike the image encoder, since spoken utterance is of variable length, the dimensions of the speech output for each encoder are not fixed. To remedy this, for each batch we take each language view and find the longest utterance in the batch. We then pad all other utterances in the respective view to be the same length as the longest. This means the sequence length of each language view is fixed inside a batch, but the sequence lengths vary across languages within a batch and across batches. Thus, letting i denote the i^{th} batch and j denote the j^{th} view, we can define max_sl_{ij} as the maximum sequence length of the utterance of the j^{th} language view in the i^{th} batch. Due to the nature of the speech encoder's CNN architecture, the sequence length of the output representation has a reduced length, but is deterministic and we can denote it $max\_sl\_out_{ij}$ for simplicity. For our dataset, $max\_sl\_out_{ij}$ typically ranges from 50 to 500. With this definition we have that the output of each speech encoder is in $\mathbb{R}^{max\_sl\_out_{ij} \times 1024}$. Where 1024 is out embedding size of the output sequence.

3.1.3 Output Pooling

As noted in the previous section, the output of each view encoder do not necessarily align across all dimensions. This creates a problem for most distance/similarity measures which typically require vector representations. Work by [6] explored a more complex set of similarity measures which utilize similarity measures across the time and location dimension, but for this work we restrict ourselves to a simplified setting of similarity measures defined only on two vectors of the same dimension. This is also more realistic in terms of retrieval when the set of target viewpoints is very large. We therefore require some sort of pooling mechanism before similarity scores can be calculated and retrieval can be performed.

Experiments in section put ref later compare pooling strategy performance empirically, but the following sections describe the basics of each mechanism.

3.1.3.1 Average Pooling

This is the simplest pooling strategy and consists of first flattening all dimensions before the last (only relevant for the image view), then averaging across the flattened dimension. One complication is the padding added to resolve the variability of sequence length in the speech encoders. Using a simple average would diminish the features of the meaningful parts of shorter sequences. Also, the sequence length is a reduced due the convolutional downsampling in each layer, which means there may be a point in the final output sequence that corresponds to both padded and unpadded input. To deal with this we only average the first avg_pool_len sequence embeddings. For the k^{th} utterance of the i^{th} batch of the j^{th} view, we denote len_{ijk} as the original

sequence length of the utterance. we calculate avg_pool_len as:

$$avg\_pool\_len = \lfloor len_{ijk} / \text{round}\Big(\frac{max\_sl_{ij}}{max\_sl\_outij}\Big) \rfloor$$

Where max_sl_{ij} and $max\_sl\_out_{ij}$ are defined in the same manner as in section 3.1.2.2 and round$(\cdots)$ rounds to the nearest integer. Here $\frac{max\_sl_{ij}}{max\_sl\_outij}$ can be thought of as the reduction ratio from the input sequence length to the output sequence length. This calculation follows the same strategy as in (Harwath's resnet orig or VQ paper).

3.1.3.2 Multi-Head Self-Attention Pooling

Considering the simplistic nature of average pooling, we also experiment with using a self attention layer to perform the pooling. Previous work (Ohishi) found that a single headed self-attention layer was beneficial when used *before* the pooling layer (specifically between the second to last and last convolutional layers), but they still used average pooling for their final representation.

Inspired by this result and the recent proliferation and success of Transformer layers (CITEME), which uses multi-headed self-attention (MHSA), we attempt to use this as an adaptive pooling mechanism for our final representation. Specifically, we use 8 heads and an positional encoder layer similar to (CITEME)and we chose the larges $max_s l_o ut_i j$ to be the maximum sequence length for the positional encoder. However, unlike the original transformer block, we do not scale the input by $\sqrt{d_{model}}$ and instead scale the positional encoding by $\frac{1}{\sqrt{d_{model}}}$ as we found this produced better results. We hypothesize that without the layer norm used after the MHSA layer in the transformer block, the original scaling gives final representations an overly large

L2 norm which negatively effects our similarity based losses. We also note, though, that adding a layer norm layer and a residual connection as in the original transformer block did not resolve this issue. Further inquiry would be required to fully explain this behavior.

After the positional encoding, we apply a dropout layer to impose a layer specific hyperparameter we can use to counteract the potential for MHSA to overfit to the dataset.

Another implementation note is that we chose to prepend an additional learnable embedding to each flattened/sequence dimension to act in the same capacity as the CLS token used in the transformer block. We then use the first embedding (which corresponds to the position of prepended embedding) as our final representation. We also experimented with simply using the first embedding of the sequence and the results of this are discussed in section 3.1.9

As a final note, we use sequence masking to prevent the final representation from attending to the embeddings corresponding to padding. And finally, we used the PyTorch implementation of MHSA in our experiments.

3.1.3.3 Transformer Pooling

We also experiment with using a full Transformer block and using a single embedding to serve as the final representation. We used the same settings and adaptations as described in 3.1.3.2 regarding number of heads, positional encoding, prepending a learned embedding, and masking. We chose to use 2048 as the internal feed forward dimension, which is the same as the original Transformer.

3.1.4 Training

We use the Adam optimization algorithm in all of the following experiments. The optimizer settings can be found in figure make this later. We also use a batch size of 128 and a learning rate of .001 with a scheduler. This scheduler enforces a linear warmup schedule (from 0 to .001) through the first 10% of training steps, and then decreases by a factor of .99 after every 50 steps.

In this chapter's experiments, all training is done using the full-graph framework of multi-view training as described in Section **??**. This is to simplify experimental variables, such as choosing which view will be the anchor, as well as to give easy access to all information shared between the views. The next chapter will explore ways to reduce the complexity of full pairwise comparisons.

Importantly, all loss functions under consideration have some component designed to encourage dissimilitude. Without this, predictions can find a trivial local solution by collapsing on constant value for all representations. Contrastive loss functions use 'negative examples' to serve this role where as loss functions like the hyperspheric loss contain the uniformity objective. In either case, we supply the non-positive examples in each batch to these sub-components.

To be clear, InfoNCE, Triplet loss, Masked Margin loss, and Scheduled Masked Margin loss all draw their negative examples from within their respective batch. The Hyperspheric loss additionally draws the samples for it's uniformity loss term from the examples in the batch. This strategy is in contrast to updating a memory-bank mechanism (CITEME)in which a much larger number of stale representations are

used and periodically updated.

3.1.5 Testing and Reporting Metrics

During inference, for each pair of views, weeach viewpoint in each view in the evaluation set is measured against all viewpoints in each other view in the evaluation set. To clarify, if N be the fix the previous sentence. Merge with explanation below

We report the recall at k scores with $k = 1, 5$, and 10, which indicates how often the target viewpoint was in the top k closest viewpoints. Also, because of space concerns and since are primarily interested in overall alignment in the latent space, for each view pair we report only the average recall at k scores between view pairs. That is, for each view pair two sets of recall at k scores are calculated: 1) a set in which the first view of the pair is treated as the input and the second view is the target, and 2) a set where the second of the pair is treated as the input and the first view is the target. These two directions do not necessarily produce the same results, but we did not observe a systematic of substantial deviation from the average among any view pair in any experiment in this work. We therefore chose to omit the scores for the individual directions and simply report the average.

Still, we fundamentally distinguish between view pairs between different modalities (i.e. image-speech pairings) and those withing the same modality (speech-speech). For convenience, then, we will refer to average results from speech-speech retrieval scores simply as 'cross-lingual retrieval results' and average results from speech-image pairings simply as 'image retrieval results', acknowledging that retrieving speech utterances from image input is not actually 'image retrieval'.

3.1.6 Loss Functions Experiments

This chapter's experiments involve insert final number distinct loss functions: InfoNCE, hyperspheric loss, triplet loss, masked margin loss, and scheduled mml. We compare performance of each directly in section 3.1.8. However, the hyperspheric loss contains many hyperparameters that are unique to it. Also, this loss function has never been applied to this task so optimal hyperparameters have not previously been studied. We therefore describe our tuning procedure in section 3.1.7.

After comparing loss functions, we also experiment with different pooling strategies in section 3.1.9 and additional potential settings for loss functions in section 3.1.9.1.

3.1.7 Hyperspheric Loss Experiments

As the hyperspheric loss is markedly different than the loss functions previously explored on this dataset, we chose to tune the uniformity sub-loss weighting. We chose this hyperparameter because it appeared to be the most impactful to the final results of the original paper (hsphere paper). There are in fact several hyperparameters available for adjustment for the hyperspheric loss, but due to time constraints we were unable to tune the others. We chose an optimization scheme of staring with the baseline value (1.0), and decreasing by .25 until performance ceased to increase. We kept all other hyperparameters constant during this tuning. In particular, we held the alignment loss weight at 1.0.

We used the encoder implementations described in 3.1.2 and used average pooling to combine the final output representations

	Unif. 1	Unif .875	Unif .75	Unif. 50
Eng.&Img.r1	0.05%	0.00%	8.30%	0.05%
Eng.&Img.r5	0.55%	0.50%	25.20%	0.35%
Eng.&Img.r10	1.40%	1.05%	36.95%	0.90%
Hin.&Img.r1	0.10%	0.10%	7.85%	0.10%
Hin.&Img.r5	0.45%	0.50%	23.10%	0.50%
Hin.&Img.r10	0.70%	1.25%	33.75%	0.90%
Jap.&Img.r1	0.10%	0.00%	9.00%	0.05%
Jap.&Img.r5	0.80%	0.40%	30.55%	0.40%
Jap.&Img.r10	1.40%	1.10%	43.65%	0.95%

Table 3.1: Hyperspheric loss w/ Average pooling

	Unif. 1	Unif .875	Unif .75	Unif. 50
Eng.&Hin._r1	0.20%	0.25%	5.90%	0.05%
Eng.&Hin._r5	1.25%	1.20%	18.00%	0.50%
Eng.&Hin._r10	2.00%	3.00%	27.05%	1.00%
Eng.&Jap._r1	0.00%	0.15%	6.90%	0.05%
Eng.&Jap._r5	0.20%	0.95%	21.30%	0.30%
Eng.&Jap._r10	0.55%	1.60%	33.95%	0.95%
Jap.&Hin._r1	0.20%	0.10%	6.40%	0.15%
Jap.&Hin._r5	0.60%	0.30%	19.30%	0.60%
Jap.&Hin._r10	0.90%	0.55%	28.95%	1.35%

Table 3.2: Hyperspheric loss w/ Average pooling

	Hyper.	InfoNCE	Triplet
Best Epoch	16	26	36
Eng.&Img._r1	8.30%	18.05%	16.40%
Eng.&Img._r5	25.20%	41.55%	39.20%
Eng.&Img._r10	36.95%	54.55%	51.55%
Hin.&Img._r1	7.85%	14.70%	12.95%
Hin.&Img._r5	23.10%	36.10%	32.65%
Hin.&Img._r10	33.75%	45.85%	43.55%
Jap.&Img._r1	9.00%	28.00%	22.70%
Jap.&Img._r5	30.55%	59.20%	53.15%
Jap.&Img._r10	43.65%	72.35%	67.05%

Table 3.3: Objective comparison w/ average pooling on image retrieval

The results for the image-audio (image retrieval) view pairings can be found in figure INSERT LATERand the audio-audio pairs (aka cross-lingual) shown in figure INSERT LATER. As can be seen in both image retrieval and cross-lingual settings, the overall performance of the hypersheric loss is highly dependent on the weighting of the uniformity sub-loss. In all following experiments, we use the best performing uniformity weighting, .75.

3.1.8 Base comparison

This set of experiments compares the results of several prominent loss functions that been applied to image retrieval tasks. Specifically, we compare InfoNCE loss, triplet loss, masked margin softmax loss, adaptive mean margin softmax loss, and hyperspheric loss with the hyperparameters described in section 3.1.7. Training, inference, and evaluation procedures are as described in sections 3.1.4 and 3.1.5.

	Hyper.	InfoNCE	Triplet
Best Epoch	16	26	36
Jap.&Eng._r1	6.90%	11.40%	8.05%
Jap.&Eng._r5	21.30%	29.00%	24.70%
Jap.&Eng._r10	33.95%	41.30%	35.85%
Hin.&Eng._r1	5.90%	11.55%	9.65%
Hin.&Eng._r5	18.00%	27.25%	23.35%
Hin.&Eng._r10	27.05%	37.55%	32.85%
Jap.&Hin._r1	6.40%	10.50%	6.30%
Jap.&Hin._r5	19.30%	27.30%	21.80%
Jap.&Hin._r10	28.95%	36.80%	30.70%

Table 3.4: Objective comparison w/ average pooling on cross-lingual retrieval

Results for image retrieval pairings can be found in Table 3.3 and cross-lingual results can be found in Table 3.4. We highlight several findings from this experiment:

1. InfoNCE loss consistently outperforms all other losses in both image retrieval and cross-lingual settings.

2. Cross-lingual retrieval appears more difficult than image retrieval for all losses.

3. Japanese and image representations appear remarkably strongly aligned for both InfoNCE and triplet loss.

Finding 1 provides empirical evidence that InfoNCE outperforms a broad range of loss functions. Experiments by [17] showed that MMSM outperformed the form of triplet loss in our experiments, but our finding provide evidence that InfoNCE outperforms a broader range of loss functions.

	Lr .001 Drop .1	Lr .0001 Drop .1	Lr .0001 Drop .3	Lr .0001 Drop .5
Best Epoch	1	15	21	32
eng&img_r1	0.25%	14.10%	11.05%	4.55%
eng&img_r5	0.70%	34.20%	33.80%	14.15%
eng&img_r10	1.55%	45.70%	44.85%	23.50%
hin&img_r1	0.05%	9.70%	9.05%	3.65%
hin&img_r5	1.15%	27.40%	25.35%	13.60%
hin&img_r10	1.75%	37.70%	34.65%	21.25%
jap&img_r1	0.15%	18.70%	19.10%	6.20%
jap&img_r5	1.10%	46.85%	45.35%	20.85%
jap&img_r10	2.10%	62.15%	59.45%	30.05%

Table 3.5: Multi-head attention pooling using InfoNCE loss. Image retrieval results

Finding 2 and 3 also aligns with the general findings of [17], and our experiments show that this trend continues for InfoNCE. Considering the strong performance, all remaining experiments will use InfoNCE loss.

3.1.9 Pooling Experiments

As described in 3.1.3, there are many different strategies to pool the final output representations of each view. Since neither MHSA or a Transformer layer has been used a pooling strategy, we first attempt to tune hyperparameters of the simpler of the two layers, the MHSA. We chose to adjust the learning rate and the dropout percentage. Due to time constraints we were not able to fully tune these parameters, but we were able to explore an array of settings. The image retrieval scores for these settings can be found in Table 3.5 and the cross-lingual scores can be found in Table 3.5.

	Lr .001 Drop .1	Lr .0001 Drop .1	Lr .0001 Drop .3	Lr .0001 Drop .5
Best Epoch	1	15	21	32
hin&eng_r1	0.20%	7.60%	6.50%	2.20%
hin&eng_r5	1.65%	20.85%	18.80%	9.25%
hin&eng_r10	2.70%	29.20%	27.80%	14.80%
jap&eng_r1	0.15%	6.70%	6.25%	2.05%
jap&eng_r5	0.70%	19.80%	19.05%	7.95%
jap&eng_r10	1.05%	29.30%	26.90%	12.90%
jap&hin_r1	0.20%	5.75%	5.00%	1.85%
jap&hin_r5	0.80%	17.15%	15.65%	7.85%
jap&hin_r10	1.45%	26.50%	22.40%	12.60%

Table 3.6: Multi-head attention pooling using InfoNCE loss. Cross-lingual retieval results.

Comparing these results with those for average-pooled InfoNCE, MHSA is clearly underperforming. We first found that the original learning rate of .001 made learning unstable and decreasing to .0001 helped considerably. We next hypothesized that the lower performance might be due to some amount of overfitting of the attention mechanism. This prompted us to adjust the percentage of dropout applied after the positional encoding layer. Our results contradict this hypothesis as it seems increasing regularization did not improve generalization.

3.1.9.1 Additional settings

These above results prompted us to try several other uses of MHSA as well as the Transformer block. First, we tried to simply use the first embedding of the flattened sequence instead of the additional learned prepended token. We also tried

	Mh Attn. Pool No CLS	Avg. Pool Int. Mh Attn.	Transformer Lr .0001 Drop. .1
Best Epoch	21	33	50
Eng.&Img.r1	9.55%	1.15%	0.30%
Eng.&Img.r5	27.85%	4.30%	1.10%
Eng.&Img.r10	38.50%	7.90%	2.25%
Hin.&Img.r1	7.90%	1.10%	0.25%
Hin.&Img.r5	23.75%	3.90%	0.90%
Hin.&Img.r10	32.55%	6.50%	2.15%
Jap.&Img.r1	16.55%	3.40%	0.35%
Jap.&Img.r5	43.15%	11.15%	2.05%
Jap.&Img.r10	57.50%	18.85%	3.35%

Table 3.7: Additional configurations for InfoNCE loss.

to place the MHSA between the second to last and last layer and simply feed that to the last convolutional layer to then get average-pooled. This is a similar strategy of that explored by [17], but we maintain our 8 head configuration rather than using a single head. The learning rate for this setting was returned to .001 (since that worked well previously for average pooling) and dropout the dropout kept at .1. Finally we try the Transformer block, which has an additional layer norm layer and two linear layers. Results for image retrieval can be found in Table 3.7, we omit the cross-lingual results for space concerns and redundat findings.

The recal scores indicate that removing the prepended token has a marginally negative impact on performance. Somewhat surprisingly, though, the internal MHSA configuration and the Transformer block proved to subtantially harm retrieval. This may be due to a lack of robust tuning, but it should be noted that average pooling has not hyperparameters itself and does not require tuning.

Chapter 4

Loss Complexity

Previous work in this area has noted that fully optimizing retrieval performance requires the tuning of each direction of each pair of views being optimized (Ask Dr. Harwath). Since the number of pairings grows quadratically with each additional view, this can result in an untennable number of parameters to tune. Furthermore, work by Chen et al. [1] and He et al. [8] have noted that state-of-the-art contrastive loss objectives are sensitive to the number of negative examples. Since negative samples are commonly taken from within each mini-batch [18, 11], this means the best performing models need to be trained either using large batch sizes or a memory-bank mechanism with stale representations [8] check that this is the correct citation. It is as yet unclear how the increase in number of views/modalities effects this need for negative examples. We hypothesize that as the number of additional views are added, all of which contain some aspect of the underlying latent information, the number of necessary negative examples might decrease. These ideas motivate this chapter's experiments in which we explore strategies reduce the growth to a linear increase and assess the impact on performance.

We start with our experimental design in Section 4.1, then we discuss our results in Section 4.2.

4.1 Experimental Design

Most aspects of our experimental setup are identical to Section 3.1, including the dataset and encoder architectures. Again each view has a dedicated encoder and final representations are obtained by average pooling. We use same Adam optimizer with a learning rate of .001 and InfoNCE loss.

4.1.0.1 Experimental Variables

Our main experimental variables involve the multi-view training framework outlined in Section **??**. Instead of using all view-pairs (i.e. the *full-graph*), we try three variations of the *anchor* framework.

In the first we simply use the image view as the anchor. That is, we only compute the InfoNCE loss from view pairs that include the image view. As the only view in the visual modality, this seems a natural choice.

In the second variation, we use an 'average view' as the anchor, which is an average representation of all views. Specifically, for each datapoint we average the final representations of each viewpoint (recall our definition of datapoint and viewpoint from Section 2.1). This effectively creates a centroid for each datapoint. Within the InfoNCE loss, our intuition is that positively associated viewpoints will be drawn closer to their own centroid, while being pushed away from centroids of other datapoints.

In the third variation, we deviate slightly from the multiview contrastive framework of (CITEME)and use an adapting the average view as our anchor. This adapting

Image Ret.	Full-Graph	Image Anc.	Avg. Anc.	Cont. Others
Best Epoch	26	23	37	16
E&I.avgR1	18.05%	19.55%	0.10%	13.05%
H&I.avgR1	14.70%	16.55%	0.00%	12.30%
J&I.avgR1	28.00%	26.30%	0.05%	25.20%
E&I.avgR5	41.55%	44.00%	0.45%	31.80%
H&I.avgR5	36.10%	37.85%	0.40%	25.90%
J&I.avgR5	59.20%	58.90%	0.35%	55.55%
E&I.avgR10	54.55%	57.65%	0.80%	44.55%
H&I.avgR10	45.85%	47.10%	0.60%	34.60%
J&I.avgR10	72.35%	72.70%	0.75%	67.95%

Table 4.1: Loss complexity reduction strategies. Image retrieval results.

average view changes according to which view is being contrasted. To be concrete, we iterate over each view in a batch and for each view we use the average of all other views as the contrasting view for the InfoNCE loss. This differs from our second variation in that instead of a fixed full average, which will contain in it the information from any view contrasted with it, we instead remove the information from the view being contrasted.

For each of the three variations we train a new set of encoders from scratch all under the same set of conditions other than the three variations described above.

4.2 Resuts

The recall at 1, 5, and 10 results for image retrieval for these three variation can be seen in Table 4.1. The Full-Graph column is the model trained with full-graph InfoNCE loss.

Somewhat surprisingly, image retieval performance increased slightly in many instances in the image view anchor training scheme. In one sense, it would seem natural that solely focusing on the image retrieval task in the loss function should produce better image retrieval scores. However, when viewed from the lens of shared mutual information, one might expect that, at an abstract level at least, removing one language encoder's incentive to learn associations with other languages would harm the model's ability to properly encode speech into a space that is shared by all languages and images. Results from Harwath et al. [3] seemed to confirm this notion of increased cross-lingual mutual information improving image retrieval. Their experiments showed a 3-4% increase in recall at 5 scores for English and Hindi image retrieval (one direction, no averaging).

The second variation was unable to learn under it's respective setting. This is perhaps due to instability caused by having the representation of a view present in both sides of the contrasting loss. Further investigation is needed to properly explain this though.

The third variation, though, managed to learn but was consistently outperformed by the image anchor scheme.

Looking at the cross-lingual retrieval results in Figure 4.2 tells a slightly different story. Here the image anchor scheme's performance deteriorates drastically, whereas the third variation (the adaptive average) is able to maintain relatively strong performance as compared withe full-graph setting. This is an encouraging result as it indicates there may be potential to make improvements to the scheme (such as a more sophisticated averaging mechanism) that would further close the gap for all

Cross-ling	Full-Graph	Image Anc.	Avg. Anc.	Cont. Others
Best Epoch	26	16	37	23
H&E.avgR1	11.55%	0.95%	0.05%	8.45%
J&E.avgR1	11.40%	1.40%	0.15%	8.65%
J&H.avgR1	10.50%	1.15%	0.20%	7.35%
H&E.avgR5	27.25%	4.40%	0.35%	21.70%
J&E.avgR5	29.00%	4.55%	0.60%	25.15%
J&H.avgR5	27.30%	4.05%	0.45%	19.25%
H&E.avgR10	37.55%	6.85%	0.70%	30.75%
J&E.avgR10	41.30%	6.70%	1.80%	35.35%
J&H.avgR10	36.80%	6.25%	1.00%	27.80%

Table 4.2: Loss complexity reduction strategies. Cross-lingual retrieval results.

view pairs.

Chapter 5

Parameter Efficiency

Up to this point we have encoded each view's representation with it's own encoder model. This can become quite taxing on GPU memory as the number of views to be encoded increases. One might expect though, that there may be certain subtasks being performed the language encoders that are shared across languages (such as recognition of common phonemes). In this chapter we explore the potential for parameter sharing among the language encoders. Previous work has also explored this topic (google paper), but this is the first time we are aware of that has explored this effect in visually grounded settings.

5.1 Experimental Design

Most of the experimental settings are the same as described Section 4.1, however in this chapter we only use the full-graph training scheme. We use the sam Adam optimizer, a learning rate of .001, the average pooled model output as each view's final representation, and the InfoNCE loss. The only differences are outlined below.

5.1.0.1 Experimental Variables

We experiment with three variations of a shared language encoder. In the *basic* variation, we simply have one shared audio encoder and feed all language input into it.

The second and third variations are inspired by Johnson et al. [13] who prepend a language specific token to a multilingual neural translation model. Unlike [13], we do not use recurrent network, but a CNN. More importantly, though, our encoder operates on the speech signal, which contains orders of magnitude more time steps. We therefore explored two alternate options for injecting prior language knowledge into our CNN encoder.

For both of these variations, we started by appending a learned language-specific embedding to the log Mel filter bank (LMFB) spectrogram feature dimension of the input. Since each input utterance contains two dimensions/axes (the first being the number of time steps and the second the LMFBs), this means we concatenate the language embedding to the second dimension. This required expanding the filter sizes of the first convolutional layer. We chose this strategy as appending the language embedding to the input feature dimension seemed a natural choice. We chose to use a small eight dimension embedding for all the langauge embedding.

Next we applied additional layer-specific and language-specific embeddings to each of the intermediate representations between the four residual blocks of the encoder. However, the decision over which dimension/axis to append the layer and language specific embedding not as clear-cut as in the input. This is because the

meaning of each dimension cannot be so clearly interpreted. Thus, the remaining two variations we explore in our experiments involve 1) appending the language embedding to the first dimension (after the batch) and 2) appending the embedding to the last dimension. We refer to these two variations as the *channel embedding* and *sequence embedding* respectively.

Note that our encoder's 1D convolutions after the first are implemented using 2D convolutions and kernels with a 1 in the height dimension. Hence, when we refer to the 'first' dimension this is typically called the channel dimension in conventional CNN nomenclature. Likewise, the 'second' dimension refers to what is typically called the 'width' dimension and plays the role of the current downsampled sequence length. The 'height' dimension is one for all intermediate representations.

Since the channel embedding variation changes the effective number of features used (and necessarily then increases the filter sizes of all intermediate layers), it has a larger number of parameters than the sequence embedding variation.

5.2 Results

The recall at $K = 1, 5$, and 10 results for image and cross-lingual retrieval can be seen in Table 5.1.

Most noticeably, all shared encoders struggled to compete with the fully parameterized model. Surprisingly, though, the basic shared model outperformed both the channel and sequence embedding variations. Due to time constraints we were not able to investigate this further, but we find it counter-intuitive that removing param-

	Full	Shared: Basic	Seq. Emb.	Chan. Emb.
Best Epoch	26	19	17	11
Image Ret.				
E&I.avgR1	18.05%	11.35%	7.35%	2.30%
H&I.avgR1	14.70%	7.40%	4.40%	2.20%
J&I.avgR1	28.00%	8.60%	4.60%	2.25%
E&I.avgR5	41.55%	31.05%	22.50%	8.40%
H&I.avgR5	36.10%	21.20%	14.10%	8.65%
J&I.avgR5	59.20%	25.85%	16.45%	7.60%
E&I.avgR10	54.55%	42.70%	32.10%	12.80%
H&I.avgR10	45.85%	29.70%	20.20%	13.45%
J&I.avgR10	72.35%	35.60%	23.20%	12.25%
Cross-Ling				
H&E.avgR1	11.55%	2.25%	1.55%	0.20%
J&E.avgR1	11.40%	2.80%	1.75%	0.30%
J&H.avgR1	10.50%	1.35%	1.50%	0.15%
H&E.avgR5	27.25%	8.35%	6.85%	1.55%
J&E.avgR5	29.00%	10.05%	6.60%	1.50%
J&H.avgR5	27.30%	5.80%	5.45%	1.20%
H&E.avgR10	37.55%	14.10%	11.00%	3.10%
J&E.avgR10	41.30%	15.45%	10.00%	2.95%
J&H.avgR10	36.80%	10.40%	8.45%	1.65%

Table 5.1: Parameter efficiency strategies. One encoder for all languages. Cross-lingual retrieval resutls.

eters and *a-priori* information about the input language would harm performance in this manner.

Another surprising result is that the sequence embedding variation performed much better than the channel embedding variation despite having more parameters. Recall that the channel dimension is effectively the feature dimension here.

Appendices

Appendix A

Lerma's Appendix

The source LaTeX file for this document is no longer quoted in its entirety in the output document. A LaTeX file can include its own source by using the command `\verbatiminput{\jobname}`.

Bibliography

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[2] M. J. Bertin et al. *Pisot and Salem Numbers*. user Verlag, Berlin, 1992.

[3] David Harwath, Galen Chuang, and James Glass. Vision as an interlingua: Learning multilingual semantic embeddings of untranscribed speech. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4969–4973. IEEE, 2018.

[4] David Harwath and James Glass. Deep multimodal semantic embeddings for speech and images. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 237–244, 2015.

[5] David Harwath, Wei-Ning Hsu, and James Glass. Learning hierarchical discrete linguistic units from visually-grounded speech. *arXiv preprint arXiv:1911.09602*, 2019.

[6] David Harwath, Adria Recasens, Dídac Surís, Galen Chuang, Antonio Torralba, and James Glass. Jointly discovering visual objects and spoken words from raw sensory input. In *Proceedings of the European conference on computer vision (ECCV)*, pages 649–665, 2018.

[7] David Harwath, Antonio Torralba, and James R Glass. Unsupervised learning of spoken language with visual context. *Neural Information Processing Systems Foundation, Inc.*, 2017.

[8] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[11] Gabriel Ilharco, Yuan Zhang, and Jason Baldridge. Large-scale representation learning from visually grounded untranscribed speech. *arXiv preprint arXiv:1909.08782*, 2019.

[12] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous. Unsupervised learning of semantic audio representations. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 126–130. IEEE, 2018.

[13] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

[14] Donald K. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[15] Leslie Lamport. *LATEX: A document preparation system*. Addison-Wesley, 2nd edition, 1994.

[16] F Mittelbach M Goosens and A Samarin. *The LATEXCompanion*. Addison-Wesley, 1994.

[17] Yasunori Ohishi, Akisato Kimura, Takahito Kawanishi, Kunio Kashino, David Harwath, and James Glass. Trilingual semantic embeddings of visually grounded speech with self-attention mechanisms. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4352–4356. IEEE, 2020.

[18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[19] Deb Roy. Learning from sights and sounds: a computational model. *PhD Thesis, MIT Media Laboratory*, 1999.

[20] Michael Spivak. *The joy of TEX*. American Mathematical Society, Providence, R.I., 2nd edition, 1990.

[21] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[22] Alf J. van der Poorten. Some problems of recurrent interest. Technical Report 81-0037, School of Mathematics and Physics, Macquarie University, North Ryde, Australia 2113, August 1981.

[23] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

Index

Vita

Craig William McCluskey was born in Minneapolis, Minnesota on 20 May 1950, the son of Dr. William R. McCluskey and Lucilla W. McCluskey. He received the Bachelor of Science degree in Engineering from the California Institute of Technology and was commissioned an Officer in the United States Air Force in 1971. He entered active duty in October, 1971, and was stationed in Denver, Colorado, Colorado Springs, Colorado, Panama City, Florida, and Sacramento, California. He separated from the USAF in 1975 and worked as an engineer for several small electronics companies in California before moving to Colorado Springs, Colorado to work for Hewlett-Packard in 1979. He left Hewlett-Packard in 1989 and joined a small company based in Herndon, Virginia, working out of his house as a "remote" engineer designing parts of the Alexis satellite for Los Alamos National Laboratories. Laid off when his portion of the satellite was completed, he applied to the University of Texas at Austin for enrollment in their physics program. He was accepted and started graduate studies in August, 1991.

Permanent address: 110 Charles St
Rockville, Maryland 20850

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.