Multi-domain adversarial training of neural network acoustic models for distant speech recognition

Seyedmahdad Mirsamadi and John H.L. Hansen

Center for Robust Speech Systems (CRSS)
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas, Richardson, Texas, USA
{mirsamadi, john.hansen}@utdallas.edu

Abstract
Building deep neural network acoustic models directly based on far-field speech from multiple recording environments with different acoustic properties is an increasingly popular approach to address the problem of distant speech recognition. The currently common approach to building such multi-condition (multi-domain) models is to compile available data from all different environments into a single train set, discarding information regarding the specific environment to which each utterance belongs. We propose a novel strategy for training neural network acoustic models based on adversarial training which makes use of environment labels during training. By adjusting the parameters of the initial layers of the network adversarially with respect to a domain classifier trained to recognize the recording environments, we enforce better invariance to the diversity of recording conditions. We provide a motivating study on the mechanism by which a deep network learns environmental invariance, and discuss some relations with existing approaches for improving the robustness of DNN models. The proposed multi-domain adversarial training is evaluated on an end-to-end speech recognition task based on the AMI meeting corpus, achieving a relative character error rate reduction of +3.3% with respect to a conventional multi-condition trained baseline and +25.4% with respect to a clean-trained baseline.

Keywords: Distant Speech Recognition, far-field microphone, Recurrent Neural Network, Adversarial Training, Multi-domain Speech Data

1. Introduction

Recurrent Neural Networks (RNN) have quickly gained considerable attention as an attractive alternative to conventional hybrids of Deep Neural Networks (DNN) and Hidden Markov Models (HMM) for acoustic modeling in
Automatic Speech Recognition (ASR). They achieve competitive accuracies in many Large Vocabulary Continuous Speech Recognition (LVCSR) tasks [1, 2, 3], while resulting in a drastically simplified overall ASR pipeline compared to HMM-based models. Along with this change, there has been a shift of focus in strategies to address the problem of distant (far-field) speech recognition, where speech is recorded using a far-field microphone and thus corrupted by room reverberation and environmental noise. The variability of the exact subject location, as well as microphone position, introduces almost an unlimited number of possible reverberation/noise combinations that could be employed for training acoustic models. Traditionally, the problem of far-field ASR has been addressed by front-end enhancement strategies that are designed to compensate for the effects of noise and reverberation. This includes the use of microphone array front-ends which employ beamforming to reduce far-field distortions [4, 5, 6], and single-channel enhancement techniques which aim at recovering clean speech features from noisy and reverberant observations [7, 8, 9].

The goal in these enhancement frameworks is to reduce the environment variabilities so as to facilitate the use of an acoustic model trained on clean speech. In contrast to these conventional approaches, the use of deep learning in ASR has popularized an alternative solution which addresses the far-field problem from an acoustic modeling perspective, directly using reverberant features without any manually-designed enhancement pipelines. By training models on a large corpus of far-field speech from different rooms with different acoustic properties, the network learns to perform the necessary feature transformations within the hidden layers to compensate for far-field distortions, resulting in robust final-layer representations that are less impacted by recording conditions. Assuming there is sufficient diversity in the train data in terms of reverberation times, direct-to-reverberation ratios, etc., the resulting model is expected to generalize to unseen recording conditions.

Modeling the feature distributions of speech phonemes based on reverberant data is a difficult learning problem due to the long-term context-dependent distortions introduced by reverberation, which span multiple frames [7]. This makes recurrent networks an attractive choice for far-field acoustic modeling. The ability of RNNs to remember relevant past information enables them to sufficiently model the long-term correlations in reverberant speech. The resulting model learns to compensate environmental variabilities in the deeper representations. As a result, the current focus in far-field ASR is on more effective DNN/RNN architectures and improved learning algorithms that can automatically derive useful representations directly based on far-field data. The studies in [10] and [11] introduce modifications to the standard Long Short Term Memory (LSTM) networks in order to improve information flow in time and through the network layers. They report improvements in ASR accuracy by using these alternative recurrent structures. Other studies attempt to improve the model by extracting auxiliary features which describe the recording condition and appending them to the spectral features [12, 13, 14]. These factor-aware approaches are expected to guide the training procedure by providing additional information about the recording environment. Another group of approaches make use
Figure 1: (a) Conventional multi-condition training: far-field data from different rooms is combined into a single train set (b) Multi-domain adversarial training: room labels are used during training to achieve improved invariance to recording conditions.

of parallel clean data to improve training. They show that if a training dataset consisting of pairs of close-talk and far-field utterances is available, the model training can be guided to jointly learn both enhancement and recognition.

The current common way to build multi-condition acoustic models is to compile speech data from different recording environments (e.g., different rooms with different reverberation times) into a single train set, and then build models based on this combined dataset, ignoring the environment labels during training (Fig. 1(a)). This is similar to the early work on multi-style training for robust speech recognition under stress. In the far-field scenario, by providing many training examples from different environments, the network is expected to derive robust representations that are invariant with respect to the range of recording conditions. Ideally, the recording environment of an utterance should be indistinguishable from the features of the last hidden layer. However, as will be shown in Section 4.2, in practice there is sustained residual information concerning the recording environment at the last hidden layer.

The goal of this study is to improve multi-condition training of RNN acoustic models by incorporating knowledge about the recording environment of each utterance into the training process (Fig. 1(b)). We assume a dataset of far-field speech from multiple different rooms is available, in which each utterance is labeled not only by a phoneme or character sequence, but also by an additional label that indicates which room it belongs to. The intention is to use this available corresponding meta-data about the training utterances in order to force better invariance with respect to recording conditions. To achieve this, we
propose to use adversarial training with respect to a domain recognizer that is built on top of the hidden layers in the network, and is expected to predict the environment label of an input utterance. We refer to the different rooms and recording conditions as multiple domains within the train data which exhibit slightly different distributions. In this context, we use multi-domain training in place of multi-condition training to emphasize the use of domain labels during training.

The idea of adversarial training was first proposed in [21] as a generative model that could learn the data distribution by explicitly trying to make samples from the modeled distribution indistinguishable from actual training examples. Generative Adversarial Networks (GANs) were used in [21] for the task of realistic image generation. Since then, GANs have been successfully used in many different tasks and applications where invariance is needed with respect to different distributions. For example, the works in [22] and [23] use adversarial training for effective transfer learning by making use of an unlabeled adaptation set from a target distribution. Also, the study in [24] uses a Gradient Reversal Layer (GRL) similar to [23] in order to improve noise robustness of hybrid DNN-HMM models by negating the gradients derived from a noise-type classifier.

In this study, we use adversarial training to achieve better invariance with respect to the different rooms in a dataset of far-field speech. We extend domain-adversarial training to be useful in far-field ASR by generalizing it to multiple domains and to recurrent networks. It will be shown that by training the input layers of the RNN-CTC models adversarially to the operation of a domain classifier, we can enforce better invariance among the different domains and improve recognition accuracy of the resulting model. To facilitate the use of adversarial training with multiple data domains, we propose to use the KL divergence between network predictions and a uniform distribution as an adversarial objective, instead of the gradient reversal approach commonly used with binary domains [23, 24]. Moreover, we propose to train the network in an alternating fashion (optimizing the parameters of the discriminator while keeping the generator fixed, and vice versa), instead of the typical joint training achieved by gradient reversal. This provides a more stable training for our RNN-CTC models. Connections from the introduced multi-domain adversarial training framework will be explained to some of the other existing approaches such as factor-aware training [12] and multi-task learning [25], which better demonstrate the role of adversarial training in achieving environmental robustness.

The remainder of this paper is organized as follows. Section 2 describes the signal model for distant speech in the form of an approximate relationship between spectral features of far-field and close-talking speech. Section 3 reviews RNN-CTC acoustic modeling, which is used throughout this study as the main architecture based on which the proposed approach is developed and evaluated. Section 4 provides an analysis of the mechanism by which a deep network learns to compensate for environment variabilities. This study reveals an opportunity for using adversarial training in order to improve model robustness. In Section 5 we introduce the proposed multi-domain adversarial framework.
for training RNN-CTC models, and compare it with some existing approaches for improving far-field robustness. Section 6 provides experimental results on a meeting transcription task based on the AMI corpus which demonstrates the effectiveness of the proposed approach. Section 7 provides a brief summary of the study and highlights the main conclusions.

2. Distant Speech Model

The effect of reverberation is often modeled by a Room Impulse Response (RIR) which characterizes the acoustic path from the speaker’s location to the microphone. A far-field signal is represented by a convolution between the original clean speech and the RIR, on which an additive interference is imposed to represent environment noise. The RIR length is a function of room reverberation time, and is almost always much larger than the short-time window length typically used in ASR (20-30 msec). As a result, the convolutive relationship in the time domain cannot be represented by a simple multiplicative relation in the Short-time Fourier Transform (STFT) domain. Rather, far-field speech in each frequency band is represented by another convolutive relationship as follows:

\[
X(k, m) = \sum_{p=0}^{L_H-1} H(k, p)S(k, m-p) + E(k, m),
\]

where \(S(k, m), X(k, m), H(k, m)\) and \(E(k, m)\) are the time-frequency representations of the clean speech, far-field microphone signal, RIR, and the additive environment noise, respectively. Variables \(k\) and \(m\) denote frequency bin and time frame indices, and \(L_H\) is the length of the frequency-domain filter \(H(k, p)\) (the length \(L_H\) is assumed to be the same for all frequency bins). The model in (1) is an approximation of the precise relationship in the STFT domain \[27\], and is very useful to describe the distortions in far-field speech. According to \[1\], the spectral content of each frame in a far-field recording influence multiple subsequent frames due to RIR effects, creating a temporal smearing effect which causes long-term dependencies in the features used in ASR. This context-dependent long-term distortion is what makes far-field ASR a challenge. HMMs, in particular, have difficulty modeling such long-term correlations due to their inherent conditional independence assumption \[28\].

3. RNN-CTC acoustic modeling

The feedback connections in RNNs enable them to model the temporal dynamics of a feature sequence. Therefore, unlike DNN-HMM hybrids which use a feed-forward network to predict context-dependent HMM states (senones), RNNs can be used for acoustic modeling without the need for a separate mechanism to handle temporal dynamics. These end-to-end RNN models directly map the sequence of speech features to a sequence of labels (phonemes or characters), without requiring any explicit alignments. There are two major network
structures which enable such direct mapping from acoustic features to label sequences: RNN-CTC models which use Connectionist Temporal Classification (CTC) [29] for alignment, and encoder-decoder models in which a separate attention mechanism is employed to learn the alignment [30]. Although the focus of this study is on RNN-CTC models, the introduced multi-domain training strategy can be similarly applied to hybrid DNN-HMM or encoder-decoder models as well.

Given a sequence of feature vectors $X = [x_1, \ldots, x_T]$ from a speech utterance, a deep RNN applies multiple layers of nonlinear recurrent transformations of the form,

$$h_t^{(l)} = f(h_{t-1}^{(l)}, h_{t-1}^{(l-1)}, \theta^{(l)}),$$

where $h_t^{(l)}$ is the output of layer $l$ at time frame $t$ ($h_0^{(0)} = x_t$), $\theta^{(l)}$ represents the trainable parameters of layer $l$, and $f(\cdot)$ is used to generally denote the internal layer transformations of the particular recurrent architecture that is used (either a Long Short-Term Memory (LSTM) layer [31] or a Gated Recurrent Unit (GRU) layer [32]). The activations of the last hidden layer are passed to a final softmax layer of size $|S| + 1$, where $S = \{s_1, \ldots, s_{|S|}, blk\}$ is the output symbol set (phonemes or characters plus an additional blank label) and $|\cdot|$ represents the cardinality of the set. The softmax outputs at each frame are interpreted as the posterior probability of observing each of the labels at that frame:

$$p(s_{i,t} | X) = \frac{\exp(w_i^T h_t^{(L)})}{\sum_{j=1}^{|S|+1} \exp(w_j^T h_t^{(L)})}. \tag{3}$$

Here, $p(s_{i,t} | X)$ represents the probability of observing symbol $s_i$ at time $t$ given the input sequence, and $w_i^T$ denotes the transpose of a column weight vector from the softmax layer. The extra symbol (blk) represents a blank or no output at a particular frame, which enables the network to appropriately align the input features with the label sequence. Note that since the recurrent layers are often bi-directional, the posteriors at each frame are conditioned on the whole input sequence.

The CTC objective is to maximize the overall probability of the ground-truth label sequence given the observed feature sequence using any possible alignment between them. This is obtained by summing over the probabilities given by all possible alignments:

$$J_{CTC} = -\log \left( \sum_{a \in \cal A} \prod_{t=1}^T p(s_{a[t],t} | X) \right). \tag{4}$$

Here, $\cal A$ is the set of all possible alignments and $a[t]$ is one such alignment which gives a symbol index for every time frame $t$. The gradients resulting from (4) are back-propagated through time and over all hidden layers to tune the network parameters. To decode a particular test utterance, the simplest approach is to use a memoryless search by selecting the most active output at each frame followed by removing blanks and label repetitions (referred to
as best-path decoding). Alternatively, we can track multiple paths in a beam search algorithm similar to [33] to find the most likely label sequence.

4. Environmental Invariance in Hidden Representations

4.1. Environment-specific features in hidden layers

It is known that during training, the internal representations of a deep network become increasingly invariant to those variations in data which are irrelevant to the intended classification task. For an RNN acoustic model trained on multi-condition far-field data, we expect the hidden features to be less sensitive to the recording environment as we move toward the deeper layers in the network. At the final hidden layer, the discovered representations should be maximally discriminant with respect to speech phonemes, with minimum variance resulting from the recording environment. Our previous study explains how this environmental invariance is achieved using a deep network [34, 35]. Here, we extend the results with a comparison between deeper and shallower models which better reflects the mechanism that enables the network to learn environment invariance.

Fig. 2 shows the hidden representations from a 3-layer RNN-CTC model that are mapped into a 2D plane through a Linear Discriminant Analysis (LDA) projection. The network has been trained on far-field data from the AMI corpus [36], which contains speech recorded by table-top microphones in 3 different meeting rooms (more details about the AMI data is provided in Section 6.1). The input features from the different rooms are significantly overlapped in the

Figure 2: RNN hidden features projected onto the 2D plane (best viewed in color).
input feature space (Fig. 2(a)). This is expected because there is no explicit room-specific feature in the input feature vectors (only Mel filterbank coefficients which represent spectral characteristics of speech). However, it can be observed in Fig. 2(b) that in the first hidden layer, the network tries to map the features from the different rooms (domains) into separate subspaces. In other words, the first hidden transformation automatically learns to extract features that are indicative of the recording environment of an utterance. Note that this is achieved while the network receives no supervising information about the environment to which each training example belongs. The only supervision provided to the network during training is the output character sequence. The network implicitly learns that in order to accurately predict the label sequence, it first needs to project the data from different domains to different subspaces by extracting environment-specific features. The subsequent layers, however, show an increasing overlap between the data from different rooms, indicating the network’s attempt to dervie room-invariant features. The process of representation learning in multi-condition trained networks can thus be viewed as a two-step procedure. The initial layers map data from different domains into separate subspaces by extracting room-specific features, and the subsequent layers learn to use this encoded domain knowledge to compensate the differences.

The LDA analysis in Fig. 2 explicitly uses the domain information as class labels. However, since LDA is a linear transformation of the data (one that maximizes class separation), it cannot inherently change the nature of the features or obtain any nonlinear latent features with increased class separability. In other words, LDA corresponds to a change of the coordinate system, so that we view the data from an angle which offers maximum separation between classes. Therefore, LDA is an appropriate choice to study the domain invariance in the different layers of a neural network: It indicates the maximum obtainable class separability, without any nonlinear transformations that inherently change the features or transform them to uncover nonlinear manifolds which offer better separability.

4.2. domain classification accuracy in hidden layers

To quantitatively assess this propagation of domain information within the network, we employ a simple linear classifier based on features from the different layers to predict which of the three meeting rooms of the AMI corpus an input utterance belongs to. Here, we explicitly use the room labels to train a logistic regression model as a room classifier while keeping the parameters of the trained RNN-CTC model fixed (See Section 6.1 for details about the setup). The resulting accuracies are depicted in Fig. 3. The solid lines indicate the accuracy of room classification based on the hidden features of different layers, when the model is trained on far-field data. It is observed that predicting the recording rooms directly based on input filterbank features using a linear classifier results in low accuracy, since there is no feature in the input which explicitly describes room characteristics. However, using the first hidden layer representations results in a much higher accuracy for both 3-layer and 6-layer RNN models. This indicates that the network has automatically derived features in the first layer.
that bear information about the characteristics of the recording environment. The subsequent layers, however, show gradually decreasing domain classification accuracy, which indicates that the network is trying to discover domain-invariant features as we move towards the output layer. Note that these results have been obtained using a simple linear model for domain classification (logistic regression). If we instead use appropriate nonlinear feature transformations (i.e., an arbitrarily deep domain classifier), it is possible to achieve high domain accuracies based on the input filter-bank features as well. However, the goal here is to simply measure the amount of domain knowledge already encoded in the feature representations at each layer, without any domain-supervised nonlinear transformations to extract such features.

The domain classification accuracy at the first hidden layer of the deeper (6-layer) network is higher than the 3-layer network. This means that a deeper network allows for better extraction of environment-specific information in the initial layers. Furthermore, the domain accuracy at the last hidden layer of the deeper network is lower, which demonstrates the ability of the deeper network to better achieve domain-invariant final representations. However, even at the final layer of the deeper network, domain classification accuracy is still significantly higher than chance-level, indicating the presence of some residual domain-specific information at this stage. In other words, although the supervision provided by label sequences encourages the network to minimize the influence of room-related factors in the final representations, in practice it cannot achieve complete domain invariance. The proposed adversarial training framework described in Section 5 uses room labels during training to better enforce this desired invariance in the hidden representations.

It is worth noting that the different rooms in the AMI corpus differ not only in terms of the acoustic properties of the rooms, but also in terms of the speakers in each room. Therefore, by training on the Single Distant Microphone (SDM) channel, the model learns differences that result from both environment
and speaker characteristics. Based on the hidden features of the SDM model, it is not possible to know how much of the encoded domain knowledge pertains to room characteristics versus speaker differences. To quantify the role of each factor, we use a similar model trained on the close-talking channel data from each room. Fig. 3 shows the domain accuracies with a clean-trained model (dashed lines) which is trained on data from the Individual Headset Microphones (IHM). The input features in this case are still far-field SDM features, but they are passed through the IHM-trained model to produce hidden features. The IHM model is trained on close-talking signals and thus cannot learn any specific information concerning room characteristics. Thus, any domain discrimination in this case is solely due to speaker differences. The SDM model, in contrast, learns both speaker and environment differences, and thus results in higher domain classification accuracy. Based on this consideration, the difference between solid and dashed lines in Fig. 3 indicates the amount of added domain information due to room differences (beyond speaker differences).

4.3. Links to factor-aware training

A group of approaches collectively referred to as factor-aware training [12, 13, 14] attempt to improve neural network acoustic models by extracting manually designed features that are indicative of a variability factor in the signal (speaker, room, etc.) and appending them to spectral features for ASR. By having access to this extra information about the signal, the network is able to derive more robust features that compensate those variations. In the context of far-field ASR, the auxiliary features may describe distance, reverberation

Figure 4: (a) factor-aware training where manually extracted room features are appended to input feature vectors, (b) Given sufficient far-field training data, the network automatically learns to extract room features in the hidden representations. (best viewed in color)
time, DRR, spectral envelope of RIR, etc. The analysis provided in the previous section indicates that by having access to sufficient far-field speech from each room, the network is able to automatically derive room-specific features in the initial hidden layers, without relying on a separate module. In essence, factor-aware training uses input features that lie on separate subspaces for each room. Given sufficient depth, a multi-condition model trained on far-field data tries to automatically achieve a similar mapping into separate subspaces in the initial layers. This process is symbolically illustrated in Fig. 4.

5. Multidomain adversarial acoustic model training

5.1. Generative Adversarial Networks (GAN)

The basic idea in GANs is to set up a game between two learners, which are referred to as generator and discriminator. The generator’s task is to create samples from a distribution that closely resembles the training data. The role of the discriminator is to distinguish between real and fake samples, (i.e., to recognize whether a sample is an actual training example or created by the generator). The idea of GANs was originally proposed in [21] and used for the task of generating images, where the generator is expected to map from random noise to a realistic image. In the context of neural networks, adversarial training refers to optimizing the parameters from two parts of a network with opposing objective functions. In the image generation task, the discriminator is a binary neural network classifier which is trained to maximize the probability of correct decision on whether its input is supplied from the training set or provided by the generator. The generator is another network which is trained to fool the discriminator by maximizing the probability that its output representation is recognized as a real image by the discriminator. In other words, the generator is trained to yield an incorrect prediction in terms of real or fake images. As a result of this competition between the two networks, the generator will try to produce images that very closely resemble actual images in the training data, therefore making it difficult for the discriminator to distinguish real images from the generator outputs. GANs have gained rapid popularity as state-of-the-art in generative modeling in many different tasks and fields [37, 38]. The idea of adversarial training was used in [22] and [23] for unsupervised adaptation, where instead of real and fake images, the discriminator is trained to distinguish between source-domain data coming from a labeled train set and target-domain data from an unlabeled adaptation set. In [23], adversarial training of the generator is implemented by using a gradient reversal layer, which is an identity transform in the forward pass, but negates the gradients during back-propagation. The gradient reversal technique was also used in [24] for a noise-type classifier to improve noise robustness of a hybrid DNN-HMM model.

5.2. Multidomain Adversarial training of RNN-CTC models

Fig. 5 shows our proposed network architecture for improved training of RNN-CTC acoustic models using multi-domain far-field speech data. The left
path in Fig. 5 (G and W networks) is a standard RNN-CTC model as described in Section 3. It uses multiple bi-directional recurrent layers on top of the input filterbank features, followed by a softmax layer to yield posterior probabilities for each of the symbols in the character (or phoneme) set. In addition to this main path, there is a secondary network (D) branching out from one of the hidden layers. Here, D is a domain classifier, (i.e., it is expected to recognize the specific recording environment for the input utterance). The domain classifier consists of a few initial dense layers at the frame level (with parameters shared in time), which are expected to map from the hidden representations of the CTC network to a new space with features that are discriminant with respect to the recording environment. Since we need a single decision for the entire utterance, intermediate features from these frame-level layers are aggregated into an utterance-level representation via mean-pooling over all frames. Note that there is in general alternate possible ways to train such networks that are expected to map from a sequence of features to a single label. These include frame-wise training where the overall label is assigned to each and every frame (and the resulting errors back-propagated from every frame), as well as different forms of pooling in time (mean, max, final-frame, etc.). For the proposed multi-domain training approach, we have empirically verified that a simple mean pooling of the intermediate domain features over all frames yields the best results. This is in agreement with what has been observed for other sequence classification tasks for speech [39] and video [40].

Assuming that the domain classifier is based on features from the $l_G$'th recurrent layer, frame-level transformations for domain classification can be written as:

$$
\mathbf{v}_t^{(k)} = \sigma (\mathbf{R}^{(k)} \mathbf{v}_t^{(k-1)} + \mathbf{c}^{(k)}),
$$

Figure 5: Network structure for the proposed multi-domain adversarial training approach.
Table 1: Different parameters of a multi-domain RNN-CTC network and their associated costs

<table>
<thead>
<tr>
<th>parameters</th>
<th>Training objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_W$</td>
<td>$J_{CTC} = -\log \left( \sum_{a \in A} \prod_{t=1}^{T} p(s_a[t], t</td>
</tr>
<tr>
<td>$\theta_D$</td>
<td>$J_D = -\sum_{i=1}^{N_D} z_i \log(q_i)$</td>
</tr>
<tr>
<td>$\theta_G$</td>
<td>$J_G = J_{CTC} + \lambda \sum_{i=1}^{N_D} z_i \log(q_i)$</td>
</tr>
<tr>
<td></td>
<td>$J_G = J_{CTC} - \lambda \sum_{i=1}^{N_D} (1 - z_i) \log(q_i)$</td>
</tr>
<tr>
<td></td>
<td>$J_G = J_{CTC} - \lambda \sum_{i=1}^{N_D} \frac{1}{N_D} \log(q_i)$</td>
</tr>
</tbody>
</table>

where $R^{(k)}$ and $c^{(k)}$ are layer weights and biases, $\sigma(\cdot)$ represents the layer non-linearity, and $v_t^{(0)} = h_t^{(l_G)}$. The output layer of the domain classifier is a softmax layer which maps from the time-averaged domain representations to posterior probabilities for each of the domains:

$$q_i = p(d_i | X) = \frac{\exp \left( u_i^T v \right)}{\sum_{j=1}^{N_D} \exp \left( u_j^T v \right)}, \quad (6)$$

where,

$$v = \frac{1}{T} \sum_{t=1}^{T} v_t^{l_D}. \quad (7)$$

Here, $d_i$ denotes the $i$’th recording environment (domain), $N_D$ is the total number of domains in the train data, and $L_D$ is the number of layers in the domain classifier.

The parameters in the second part of the CTC network ($\theta_W$) are trained using the CTC cost in the usual way as described in Section 3. The parameters of the domain classifier ($\theta_D$) are trained to maximize the probability of correct domain prediction, (i.e., by using the cross-entropy cost between the softmax outputs and ground-truth domain labels):

$$J_D = -\sum_{i=1}^{N_D} z_i \log(q_i), \quad (8)$$

where $[z_0, \cdots, z_{N_D}]$ is a one-hot encoding of the ground-truth domain label (i.e., $z_i$ is 1 if it corresponds to the correct domain and 0 otherwise). The parameters in the shared section of the network between the CTC task and the domain classifier (i.e., $\theta_G$) are optimized to reduce both the CTC cost and an adversarial domain objective ($J_A$) which is designed to oppose a correct domain classification:

$$J_G = J_{CTC} + \lambda J_A. \quad (9)$$

The parameter $\lambda$ sets the relative importance of CTC and adversarial objectives in training the generator network, which is a hyper-parameter of the
algorithm that can be tuned based on a development dataset. There are different possible choices for the adversarial cost $J_A$ which are listed in Table 1. The simplest approach is to choose $J_A = -J_D$, which corresponds to a minimax game where the domain classifier tries to maximize the probability of the correct domain, while the generator tries to minimize it. Although the operation of GANs is often described using this objective, in practice it suffers from an early saturation of the cost when the domain classifier is providing a correct prediction, thus receiving vanishing gradients and failing to achieve the adversarial task [41]. An alternative cost to overcome this problem is to reverse the domain labels for training the generator (i.e., assigning 0 to the correct domain and 1 to the other domains):

$$J_A = - \sum_{i=1}^{N_D} (1 - z_i) \log(q_i).$$

Although this works well for the original GAN structure which uses a binary discriminator, for our multi-domain training framework, switching the labels does not lead to a well-defined objective, as it yields multiple correct domains and a single incorrect domain. To be applicable to a multi-domain scenario, we propose to use the Kullback-Liebler (KL) divergence between the discriminator outputs and a uniform distribution ($z_i = \frac{1}{N_D}, i = 1, \cdots, N_D$) as the training cost for generator parameters:

$$J_A = - \sum_{i=1}^{N_D} \frac{1}{N_D} \log(q_i).$$

(11)

Here, instead of using the hard domain labels, the domain posteriors are being compared with a set of soft targets which represent a uniform distribution over all domains. The generator is thus trained to achieve a state of maximum confusion where equal probabilities are assigned to different domains.

Many original GAN studies choose to simultaneously optimize all network parameters together according to the objectives in Table 1. This means that each mini-batch of data results in an update of $\theta_D$ in the direction of minimizing domain cost and an update of $\theta_G$ to minimize the adversarial cost. Although this is an approximation for the complete iterative training (where we alternate between discriminator and generator updates, keeping one fixed and updating the other), it has been shown to perform well in practice in image generation tasks with feed forward nets. However, to have a stable learning in our multi-domain RNN training, using an iterative optimization strategy was found to be necessary. The resulting train procedure is summarized in Algorithm 1. Each epoch of the algorithm consists of one CTC pass over the train data, followed by domain discriminator updates and generator updates. Note that Algorithm 1 uses Stochastic Gradient Descent (SGD) updates for simplicity. In practice, adaptive learning rate methods such as RMSprop [42] can be used for faster convergence.
Algorithm 1: Iterative CTC and adversarial updates

Input: Features: \( X(i) = [x_1(i), \ldots, x_T(i)] \), Label sequences: \( y(i) = [y_1(i), \ldots, y_L(i)] \), Domains: \( d(i) \)

\[ \text{while stop criterion not met do} \]
\[ \quad \text{for } k = 1, \ldots, n \text{ minibatches do} \]
\[ \quad \quad \text{Sample a mini-batch of } M \text{ examples } (X(i), y(i), d(i)). \]
\[ \quad \quad p(i) = [p_1(i), \ldots, p_s(i)] = f_W \left( f_G(X(i)) \right). \]
\[ \quad \quad J_{\text{CTC}} = \frac{1}{M} \sum_{i=1}^{M} \text{CTC}(p(i), y(i)). \]
\[ \quad \quad \theta_W \leftarrow \theta_W - \eta \nabla_{\theta_W} J_{\text{CTC}}. \]
\[ \quad \quad \theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} J_{\text{CTC}}. \]
\[ \quad \text{end} \]
\[ \quad \text{end minibatches} \]
\[ \text{end} \]
\[ \text{for } r = 1, \ldots, r_D \text{ do} \]
\[ \quad \text{for } k = 1, \ldots, n \text{ minibatches do} \]
\[ \quad \quad \text{Sample a mini-batch of } M \text{ examples } (X(i), y(i), d(i)). \]
\[ \quad \quad q(i) = [q_1(i), \ldots, q_{N_D}(i)] = f_D \left( f_G(X(i)) \right). \]
\[ \quad \quad J_D = -\lambda \frac{1}{M} \sum_{i=1}^{M} \log(q_d(i)). \]
\[ \quad \quad \theta_D \leftarrow \theta_D - \eta \nabla_{\theta_D} J_D. \]
\[ \quad \text{end} \]
\[ \text{end minibatches} \]
\[ \text{end} \]
\[ \text{for } r = 1, \ldots, r_G \text{ do} \]
\[ \quad \text{for } k = 1, \ldots, n \text{ minibatches do} \]
\[ \quad \quad \text{Sample a mini-batch of } M \text{ examples } (X(i), y(i), d(i)). \]
\[ \quad \quad J_A = -\lambda \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{N_D} \frac{1}{N_D} \log(q_j). \]
\[ \quad \quad \theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} J_A. \]
\[ \quad \text{end} \]
\[ \text{end minibatches} \]
\[ \text{end} \]
\[ \text{end} \]

5.3. Comparison with standard multi-task learning

The proposed network architecture in Fig. [5] is structurally similar to Multi-Task Learning (MTL) networks where two different classification tasks are solved based on a shared intermediate representation. However, in spite of this structural similarity, multi-domain adversarial training attempts to achieve the opposite of what is typically intended in MTL. In MTL, we have two different but related tasks to be learned from a single domain of data. The network parameters are optimized to reduce classification error for both tasks. By sharing some of the hidden transformations, the two tasks are expected to support each other. In other words, the intermediate features at the output of the shared section of the network are expected to be discriminant with respect to both tasks. In contrast, in multi-domain adversarial training, we have a single main task which is common to multiple data domains, and we need an intermediate representation which is discriminant for the main classification task but invariant with respect to the domains.
5.4. Comparison with factor-aware approaches

As discussed in Section 4.3, factor-aware approaches try to achieve environment robustness by using auxiliary features that are informative about the recording conditions. In spite of the similarity in what they try to achieve, the proposed adversarial training approach is fundamentally different from such factor-aware solutions. Factor-aware approaches need more than just the domain information, they need specifically-designed features that are informative about the recording conditions of each domain (e.g., speaker to microphone distance, or RIR envelope). In other words, factor-aware approaches require extra information to be known about the recording environment (both for train and test). The use of a simple one-hot feature vector which merely encodes domain information is not feasible in practice, because the test domain is always unknown and different from all of the domains present in the training data. That is, such a one-hot representation is only available for training data, not the test data which comes from a new and unseen domain. In contrast, while the proposed adversarial training approach uses domain labels during training, it does not require any such information at test time, nor does it require the test domain to be seen before. The domain information is only used during training to achieve a more robust model and to avoid overfitting to the training domains.

6. Experiments

6.1. ASR setup and data

We evaluate the proposed multi-domain training approach on an end-to-end ASR task based on the AMI meeting corpus [36]. The AMI corpus consists of speech recorded in three different meeting rooms, using both Individual Headset Microphones (IHM) and a microphone array placed on the meeting table. We use only a Single Distant Microphone (SDM) from the array to provide far-field speech data for train and test. The SDM channel poses two different problems for ASR: simultaneous speech and far-field distortions. To focus on the latter, we remove any utterances that contain overlapped speech regions from both train and test data, which leaves us with approximately 30 hours of train, 4 hours of development, and 4 hours of test data (we use the recommended data partitions for ASR outlined in [43]).

The input features are 24-dimensional Mel filterbank coefficients with speaker-level mean and variance normalization. These are extracted using 25 msec windows at a rate of 100 frames per second. We use frame-skipping [44] with a context window of 3 frames to reduce the required computations for RNN training.

---

1Three different data collection sites: University of Edinburgh (U.K.), IDIAP research institute (Switzerland), and the TNO Human Factors Research Institute (The Netherlands)

2This has been shown to have minimal performance effects on CTC models particularly with large datasets [44].
Table 2: Baseline Character Error Rates with clean-trained (IHM) and far-field (SDM) models

<table>
<thead>
<tr>
<th>Train Data</th>
<th>Test Data</th>
<th>features</th>
<th>CER (best-path)</th>
<th>CER (beam-search)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHM</td>
<td>IHM</td>
<td>fbank</td>
<td>32.8</td>
<td>32.2</td>
</tr>
<tr>
<td>IHM</td>
<td>SDM</td>
<td>fbank</td>
<td>63.6</td>
<td>62.9</td>
</tr>
<tr>
<td>SDM</td>
<td>SDM</td>
<td>fbank</td>
<td>49.1</td>
<td>48.5</td>
</tr>
<tr>
<td>SDM</td>
<td>SDM</td>
<td>fbank +</td>
<td>48.6</td>
<td>47.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one-hot domain</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The baseline acoustic model is a 6-layer recurrent network with bi-directional LSTM (BLSTM) layers containing 128 cells per direction, leading to 256 dimensional hidden layer representations. We choose an output space similar to [13], by having multi-character units (such as ‘l’ in tall and ’s in let’s), and by using uppercase characters to indicate the beginning of a word. This makes an output space of size 79, consisting of 78 character labels plus the blank symbol.

Network parameters are optimized using RMSprop [42] with an initial learning rate of 0.001 and a minibatch size of 20 utterances. Training epochs are stopped when no further improvement in Character Error Rate (CER) is observed on the development set. The results are reported using both a simple best-path decode strategy (choosing the most active output at each frame followed by removing blanks and repetitions), and also by a beam-search algorithm with a beam width of 10 to track multiple candidate sequences. In both cases, decoding is based only on acoustic scores from the RNN using no additional lexicon or language information.

Table 2 shows the performance of this baseline CTC network when trained on clean speech (IHM channel) and far-field speech (SDM channel). As expected, the clean-trained model results in a high error rate on the SDM test data due to the significant underlying mismatch in this case. The multi-condition model uses far-field (SDM) train data from multiple rooms and speaker-to-microphone distances within the AMI corpus, and thus provides significantly improved far-field robustness, yielding a +23% relative improvement with respect to the clean-trained model. Also shown in Table 2 is the performance of a room-aware model, where the room information is encoded as a one-hot vector appended to the filter-bank features. Note that this is only an oracle experiment which uses ground-truth domain labels both during train and test. In practice, the domain of the test utterance is unknown and different from those in the training data. The improvement from this simple additional factor is limited. We speculate that factor-aware models require an auxiliary feature that is more informative about the recording conditions, rather than a simple domain label.

\[\text{We have empirically verified that using this output space provides consistently better results compared to the alternative approach which uses dedicated space and apostrophe characters [1].}\]
### 6.2. Results on multi-domain adversarial training

Table 3 compares the performances provided by the proposed multi-domain adversarial framework when the domain discriminator branches out from different hidden layers of the network. In all of the experiments discussed here, a data domain refers to the different recording rooms in the AMI corpus, and the training data is always the SDM (far-field) channel. The domain discriminator is chosen to be a 2-layer network with a frame-level sigmoid layer of 256 nodes, followed by a mean-pool operation across all frames of the utterance and a final softmax layer to provide probabilities of belonging to each of the three AMI meeting rooms. The network is trained according to Algorithm 1 with \( \lambda = 20.0 \) and \( r_D = r_G = 4 \) (i.e., each epoch consists of one CTC pass to tune \( \theta_W \) and \( \theta_G \), followed by 4 iterations of cross-entropy updates on domain discriminator \( \theta_D \) and 4 iterations of adversarial updates on \( \theta_G \)). As observed in Table 3, the best results are obtained when the domain discriminator is based on layer-2 features, which yields a +3.3% relative improvement over a multi-condition baseline that does not use domain labels during training. The overall improvement with respect to the clean-trained baseline is +25.4% in this case.

Several observations can be made based on the results in Table 3. First, note that when the domain discriminator is based on features from the first hidden layer, the resulting performance is worse than the baseline. In other words, forcing environment invariance from the very first layer harms performance. This can be understood by considering the learning mechanism described in Section 4. In the first hidden layer, the network extracts domain-specific features that are indicative of the recording conditions. The subsequent layers use this encoded domain knowledge to derive the robust final representations. Therefore, forcing the features to be domain-invariant from the very first layer interferes with this inherent operation of the network, resulting in lower overall performance. Instead, we should allow the first few layers to contain domain-specific information and only force invariance at a subsequent layer where we expect higher-level knowledge that is independent of the recording conditions.

Moreover, note that when the domain discriminator is based on 6th layer features, learning cannot converge and yields a high error rate. Although this happens only in this case with the chosen hyper-parameters described here (dis-
criminator architecture, learning rate, etc.), we have observed that other choices of hyper-parameters also lead to this phenomenon. Similar difficulties have been reported for GAN training in other applications [41]. In our case, we attribute this to the network’s failure in reaching an equilibrium between the domain discriminator and the generator layers. When the generator contains all 6 BLSTM layers (last row of Table 3), it significantly outperforms the domain classifier and does not allow it to learn any discriminating information about the domains. This poorly trained discriminator will then return gradients to the recurrent layers which cause learning to diverge completely. In other words, similar to most existing GAN applications, successful training in our case depends on a careful choice of hyper parameters which enables the network to reach an equilibrium between the generator and the domain discriminator. In essence, adversarial training is an inherently more difficult learning problem, because unlike most other learning problems which seek minimization of a single cost, here we are interested in an equilibrium between two opposing objectives.

Table 4 compares the error rates resulting from different adversarial objectives used for the shared (generator) part of the network. All of the experiments in this table use hidden features from Layer-2 as input to the domain discriminator. Using a standard MTL cost results in degraded performance compared to the CTC baseline. This is expected given the discussion in Section 5.3. A conventional MTL training (i.e. non-adversarial objective) further increases the differences between the data distributions of the different environments, thus making it more difficult for the final softmax layer to learn a robust classification. Among adversarial objectives, a simple gradient reversal (GRL) failed to provide improvement, since it suffers both from early saturation as well as confusion from multiple (non-binary) domains (see the discussion in Section 5.3). Best performance is obtained when the adversarial cost is the KL divergence between discriminator posteriors and a uniform domain distribution. Such an objective achieves a state of maximal confusion between the different domains, and is well-defined for both binary and multi-domain cases.

<table>
<thead>
<tr>
<th>Training approach</th>
<th>Generator Objective</th>
<th>CER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline CTC</td>
<td>$J_A = 0$</td>
<td>49.1</td>
</tr>
<tr>
<td>CTC + MTL</td>
<td>$J_A = -\sum_{i=1}^{N_D} z_i \log(q_i)$</td>
<td>49.3</td>
</tr>
<tr>
<td>CTC + Adversarial (GRL)</td>
<td>$J_A = \sum_{i=1}^{N_D} z_i \log(q_i)$</td>
<td>49.0</td>
</tr>
<tr>
<td>CTC + Adversarial (flipped domain labels)</td>
<td>$J_A = -\sum_{i=1}^{N_D} (1 - z_i) \log(q_i)$</td>
<td>48.8</td>
</tr>
<tr>
<td>CTC + Adversarial (KL-div with uniform domain distribution)</td>
<td>$J_A = -\sum_{i=1}^{N_D} \frac{1}{N_D} \log(q_i)$</td>
<td>47.6</td>
</tr>
</tbody>
</table>
6.3. Discussion

All of the above results were obtained using the training set of AMI corpus. A relevant question is how much of the improvements from adversarial training will hold if the training data is significantly scaled up. Our experiments on subsets of the AMI corpus suggest that in addition to the amount of data, the diversity of the training utterances in terms of the recording environment is a relevant factor. Therefore, we speculate that a large training data, if collected in a very large number of different environments with diverse acoustic properties, will make the contribution from adversarial training less significant. However, it should be noted that while it is feasible to simply collect a large dataset in a few recording environments, diversifying the collection in terms of the recording rooms is a much costlier task. Therefore, using the proposed adversarial training with a reasonable number of recording rooms provides a more practical solution to make the acoustic model robust against environment variations.

Another interesting question to ask is how much adversarial training helps with increasing feature invariance. The improved recognition accuracies in Table 3 imply better invariance, but we can directly quantify this by using a simple domain classifier similar to Section 4.2. Figure 6 compares the domain classification accuracies obtained using the hidden layer features of the baseline model versus the best adversarially trained model from Table 3. The domain classifier is a simple logistic regression similar to Section 4.2. It can be observed that adversarial training has in fact made it more difficult to predict which environment the input utterance belongs to (hence the lower classification accuracy), which indicates better invariance with respect to recording environments. Also, note that in the adversarial model studied in Fig. 6, the domain discriminator is based on features from the 2nd layer, meaning that domain invariance is enforced only after the 2nd layer of the model. Therefore, the features of the first layer exhibit a high domain accuracy (significant information about the recording environment), similar to the baseline model.

A final question to consider is whether the acoustic model improvements which were demonstrated in terms of the reduced character error rates in Table 3 can actually result in better word error rates (WER) in a practical ASR system. Since the proposed adversarial training approach exclusively focuses on improving environment robustness of the acoustic model, the best way to evaluate the resulting improvement is to use metrics which evaluate the acoustic model in isolation, such as phoneme or character error rate. Whether or not an improvement in the acoustic model results in WER gains depends on the language model used for the task, as well as on the specific phonetic or orthographic lexicon used to specify words. In our experiments, we have observed that if no lexical or language information is used (simply scoring the best-path decoded CTC output), there is a consistent WER gain provided by adversarial training, although in this case the values of WER would be high due to the lack of language information. However, WER results can be confusing if an orthographic lexicon and LM is incorporated, particularly for a rather limited-vocabulary task such as the AMI corpus. In this case, strong language priors...
can fix acoustic model errors and obscure improvements, or conversely, a small acoustic model improvement can be magnified if it is able to set the beam search on the right path and fix many subsequent word errors.

7. Conclusion

In this study, we presented a novel multi-domain training approach for neural network acoustic models which makes use of environment labels in far-field speech data in order to achieve increased invariance with respect to the recording conditions in different rooms. Unlike conventional multi-condition training which combines data from different recording environments into a single set, we consider multi-environment datasets to consist of different domains with slightly different distributions. We presented an analytic study on how a deep network learns to derive environmentally robust features solely based on label sequence supervision. It was shown that the initial layers in a deep network function as a domain separator, mapping data from different rooms into different subspaces. The subsequent layers can then use this encoded domain knowledge to derive robust final representations. This propagation of domain knowledge within the hidden layers was evaluated using a simple domain classifier trained on features from the different hidden layers, which revealed that in practice there is residual domain information even in the last hidden layer, indicating insufficient domain invariance.

Further, it was shown that if the initial layers in an RNN acoustic model are trained adversarially with respect to a domain classifier which recognizes the recording environments, we can enforce better domain invariance and hence a
more robust model. The proposed multi-domain adversarial training strategy was evaluated in an end-to-end speech recognition task based on the AMI corpus. It was shown that with a domain classifier based on features from the second hidden layer, a relative improvement of +3.3% can be achieved in character error rate compared to a multi-condition trained baseline which does not use the domain labels during training. The overall performance improvement with respect to the clean-trained baseline is +25.4%.

References


