Appendix for Estimating Generalization under Distribution Shifts via Domain-Invariant Representations

Ching-Yao Chuang [1] **Antonio Torralba** [1] **Stefanie Jegelka** [1]

A. Proofs

A.1. Proof of Lemma 3

Lemma 3. *Given a hypothesis class $\mathcal{P}$, for all $h \in \mathcal{H}$,*

$$R_T(h) \leq \underbrace{\sup_{h' \in \mathcal{P}} R_T(h, h')}_{\text{Proxy Risk}} + \underbrace{\inf_{h' \in \mathcal{P}} R_T(h')}_{\text{Bias}}. \tag{1}$$

Proof. Let $h^* = \arg\inf_{h \in \mathcal{P}} R_T(h)$, by the triangle inequality,

$$\begin{aligned} R_T(h) &\leq R_T(h, h^*) + R_T(h^*) && (2)\\ &\leq \sup_{h' \in \mathcal{P}} R_T(h, h') + \inf_{h' \in \mathcal{P}} R_T(h'). && (3) \end{aligned}$$

□

A.2. Proof of Lemma 4

Lemma 4. *Given a hypothesis class $\mathcal{P}$, for all $h \in \mathcal{H}$,*

$$\underbrace{|\sup_{h' \in \mathcal{P}} R_T(h, h') - R_T(h)|}_{\text{Estimation Error}} \leq \sup_{h' \in \mathcal{P}} R_T(h'). \tag{4}$$

Proof. First, by Lemma 3, we have

$$\sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h, h') - R_T(h) \geq - \inf_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h'). \tag{5}$$

Next, by the triangle inequality,

$$R_T(h, h') \leq R_T(h) + R_T(h'). \tag{6}$$

The above inequality holds for all $h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}$, by taking supremum on both sides of the inequality, we have

$$\begin{aligned} \sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h, h') &\leq \sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} (R_T(h) + R_T(h')) && (7)\\ &= R_T(h) + \sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h'), && (8) \end{aligned}$$

implying that

$$\sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h, h') - R_T(h) \leq \sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h'). \tag{9}$$

Since $\inf_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h') \leq \sup_{h' \in \mathcal{P}^\epsilon_{\mathcal{FG}}} R_T(h')$, combining equation (5) and equation (9) completes the proof. □

A.3. Proof of Theorem 6

Theorem 6. *For all $f \in \mathcal{F}$ and $g \in \mathcal{G}$,*

$$\begin{aligned} R_T(fg) \leq R_S(fg) &+ \underbrace{d_{\mathcal{F}\Delta\mathcal{F}}(p^g_S(Z), p^g_T(Z))}_{\text{Latent Divergence}} \\ &+ \underbrace{d_{\mathcal{F}_{\mathcal{G}\Delta\mathcal{G}}}(p_S, p_T)}_{\text{Embedding Complexity}} + \lambda_{\mathcal{FG}}(g). \end{aligned} \tag{10}$$

where $\lambda_{\mathcal{FG}}(g)$ is a variant of the best in-class joint risk:

$$\lambda_{\mathcal{FG}}(g) = \inf_{f' \in \mathcal{F}, g' \in \mathcal{G}} 2R_S(f'g) + R_S(f'g') + R_T(f'g').$$

Proof. Define f^*g^* as follows:

$$f^*g^* = \arg\inf_{f' \in \mathcal{F}, g' \in \mathcal{G}} 2R_S(f'g) + R_S(f'g') + R_T(f'g'). \tag{11}$$

By the triangle inequality,

$$\begin{aligned} R_T(fg) &\leq R_T(f^*g^*) + R_T(fg, f^*g^*) && (12)\\ &\leq R_T(f^*g^*) + R_T(fg, f^*g) + R_T(f^*g, f^*g^*). && (13) \end{aligned}$$

The second term in the R.H.S of Eq. 13 can be bounded as

$$\begin{aligned} & R_T(fg, f^*g) && (14)\\ &\leq R_S(fg, f^*g) + |R_S(fg, f^*g) - R_T(fg, f^*g)| && (15)\\ &\leq R_S(fg, f^*g) + \sup_{f, f' \in \mathcal{F}} |R_S(fg, f'g) - R_T(fg, f'g)| && (16)\\ &= R_S(fg, f^*g) + d_{\mathcal{F}\Delta\mathcal{F}}(p^g_S(Z), p^g_T(Z)) && (17)\\ &\leq R_S(fg) + R_S(f^*g) + d_{\mathcal{F}\Delta\mathcal{F}}(p^g_S(Z), p^g_T(Z)). && (18) \end{aligned}$$

[1]CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Ching-Yao Chuang <cychuang@mit.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

The last inequality follows from the triangle inequality. The third term in the R.H.S of Eq. 13 can be bounded similarly:

$$R_T(f^*g, f^*g^*) \tag{19}$$
$$\leq R_S(f^*g, f^*g^*) + |R_S(f^*g, f^*g^*) - R_T(f^*g, f^*g^*)| \tag{20}$$
$$\leq R_S(f^*g, f^*g^*) + \sup_{f\in\mathcal{F}, g, g'\in\mathcal{G}} |R_S(f'g, f'g') - R_T(f'g, f'g')| \tag{21}$$
$$= R_S(f^*g, f^*g^*) + d_{\mathcal{F}_{\mathcal{G}\Delta\mathcal{G}}}(p_S(X), p_T(X)) \tag{22}$$
$$\leq R_S(f^*g) + R_S(f^*g^*) + d_{\mathcal{F}_{\mathcal{G}\Delta\mathcal{G}}}(p_S(X), p_T(X)). \tag{23}$$

Plugging the above bounds into equation (13), we have

$$R_T(fg) \leq R_S(fg) + d_{\mathcal{F}\Delta\mathcal{F}}(p_S^g(Z), p_T^g(Z)) + d_{\mathcal{F}_{\mathcal{G}\Delta\mathcal{G}}}(p_S(X), p_T(X)) + \lambda_{\mathcal{F}\mathcal{G}}(g), \tag{24, 25}$$

where the $\lambda_{\mathcal{F}\mathcal{G}}(g)$ emerges by

$$\lambda_{\mathcal{F}\mathcal{G}}(g) = 2R_S(f^*g) + R_S(f^*g^*) + R_T(f^*g^*) \tag{26}$$
$$= \inf_{f'\in\mathcal{F}, g'\in\mathcal{G}} 2R_S(f'g) + R_S(f'g') + R_T(f'g'). \tag{27}$$

□

A.4. Proof of Proposition 7

Proposition 7. *In an N-layer feedforward neural network $h = f_i g_i \in \mathcal{F}_i\mathcal{G}_i = \mathcal{H}$, the following inequalities hold for all $i \leq j \leq N-1$:*

$$d_{\mathcal{F}_{i\mathcal{G}_i\Delta\mathcal{G}_i}}(p_S, p_T) \leq d_{\mathcal{F}_{j\mathcal{G}_j\Delta\mathcal{G}_j}}(p_S, p_T)$$
$$d_{\mathcal{F}_i\Delta\mathcal{F}_i}(p_S^{g_i}(Z), p_T^{g_i}(Z)) \geq d_{\mathcal{F}_j\Delta\mathcal{F}_j}(p_S^{g_j}(Z), p_T^{g_j}(Z)).$$

Proof. Recall that an N-layer feedforward neural network can be decomposed as $h = f_i g_i \in \mathcal{F}_i\mathcal{G}_i = \mathcal{H}$ for $i \in \{1, 2, \ldots, N-1\}$, where the embedding g_i is formed by the first layer to the i-th layer and the predictor f_i is formed by the $i+1$-th layer to the last layer. We define $q_{ij} : \mathcal{Z}_i \to \mathcal{Z}_j$ to be the function formed by the i-th layer to j-th layer, where $\mathcal{Z}_k$ is the latent space formed by the encoder $g_k : \mathcal{X} \to \mathcal{Z}_k$. We denote the class of q_{ij} as $\mathcal{Q}_{ij}$.

We now prove the first inequality. By the definition of the $\mathcal{F}_{\mathcal{G}\Delta\mathcal{G}}$-divergence, for every $i \leq j$

$$d_{\mathcal{F}_{i\mathcal{G}_i\Delta\mathcal{G}_i}}(p_S, p_T) \tag{28}$$
$$= \sup_{\substack{f\in\mathcal{F}_i \\ g,g'\in\mathcal{G}_i}} |R_S(fg, fg') - R_T(fg, fg')| \tag{29}$$
$$= \sup_{\substack{f\in\mathcal{F}_j, q\in\mathcal{Q}_{ij} \\ g,g'\in\mathcal{G}_i}} |R_S(fqg, fqg') - R_T(fqg, fqg')| \tag{30}$$
$$\leq \sup_{\substack{f\in\mathcal{F}_j \\ q,q'\in\mathcal{Q}_{ij} \\ g,g'\in\mathcal{G}_i}} |R_S(fqg, fq'g') - R_T(fqg, fq'g')| \tag{31}$$
$$= \sup_{\substack{f\in\mathcal{F}_j \\ g,g'\in\mathcal{G}_j}} |R_S(fg, fg') - R_T(fg, fg')| \tag{32}$$
$$= d_{\mathcal{F}_{j\mathcal{G}_j\Delta\mathcal{G}_j}}(p_S, p_T) \tag{33}$$

The second inequality can be proved similarly. By the definition of the $\mathcal{F}\Delta\mathcal{F}$-divergence, for every $i \leq j$

$$d_{\mathcal{F}_j\Delta\mathcal{F}_j}(p_S^{g_j}(Z), p_T^{g_j}(Z)) \tag{34}$$
$$= \sup_{f,f'\in\mathcal{F}_j} |R_S(fg_j, f'g_j) - R_T(fg_j, f'g_j)| \tag{35}$$
$$= \sup_{f,f'\in\mathcal{F}_j} |R_S(fq_{ij}g_i, f'q_{ij}g_i) - R_T(fq_{ij}g_i, f'q_{ij}g_i)| \tag{36}$$
$$\leq \sup_{\substack{q\in\mathcal{Q}_{ij} \\ f,f'\in\mathcal{F}_j}} |R_S(fqg_i, f'qg_i) - R_T(fqg_i, f'qg_i)| \tag{37}$$
$$\leq \sup_{\substack{q,q'\in\mathcal{Q}_{ij} \\ f,f'\in\mathcal{F}_j}} |R_S(fqg_i, f'q'g_i) - R_T(fqg_i, f'q'g_i)| \tag{38}$$
$$= \sup_{f,f'\in\mathcal{F}_i} |R_S(fg_i, f'g_i) - R_T(fg_i, f'g_i)| \tag{39}$$
$$= d_{\mathcal{F}_i\Delta\mathcal{F}_i}(p_S^{g_i}(Z), p_T^{g_i}(Z)) \tag{40}$$

□

B. Experiments

B.1. The Role of Inductive Bias

Besides the number of layers and number of hidden neurons, we investigate the importance of inductive bias for domain invariant representations, by replacing the CNN encoder with an MLP. The results for M→M-M are shown in Figure 1. The target error with the MLP encoder is significantly higher than with a CNN encoder. Even

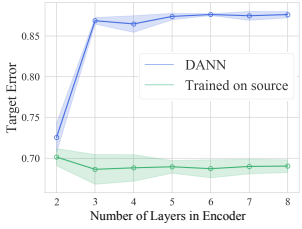
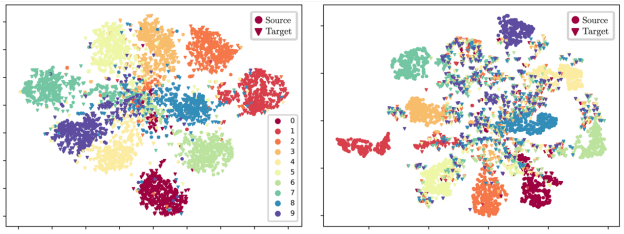
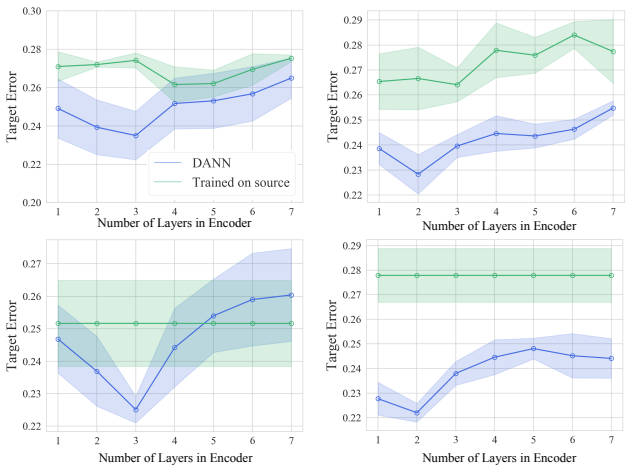


Figure 1. The role of inductive bias: DANN with fully connected layers instead of a CNN.



(a) t-SNE (CNN) (b) t-SNE (MLP)

Figure 2. t-SNE projections of representations with different inductive biases. CNN encoders result in source and target representations that are well aligned. In contrast, MLP encoders lose label-consistency while minimizing the latent divergence between domains.



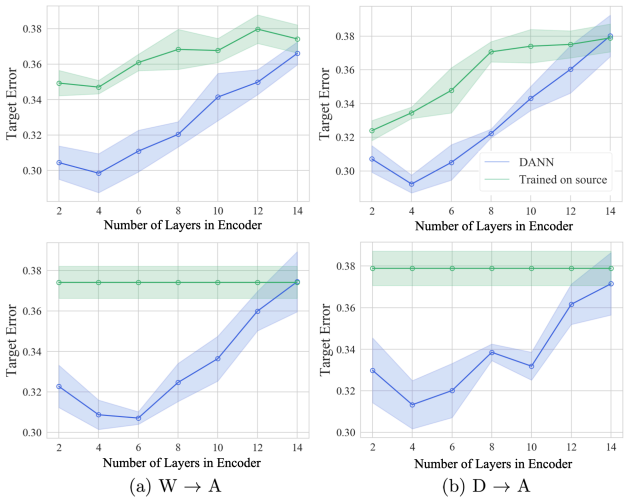
(a) B → K (b) D → B

Figure 3. Amazon reviews dataset. First row: Fixed predictor class, varying number of layers in the encoder. Second row: Fixed total number of layers and optimizing domain-invariant loss in a single intermediate layer.

more, with respect to target error, the model is very sensitive to its complexity. Increasing the number of layers worsens the performance, and is worse than not doing any domain adaptation. To gain deeper insight, in Figure 2, we use a t-SNE (Maaten & Hinton (2008)) projection to visualize source and target distributions in the latent space. With the inductive bias of CNNs, the representations of target domain well align with those in the source domain. However, despite the overlap between source and target domains, the MLP encoder results in serious label-mismatch.

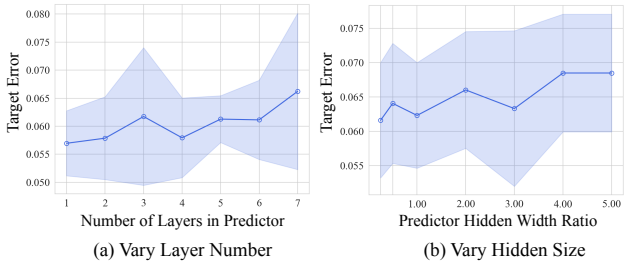
B.2. Effect of Embedding Complexity

Sentiment Classification In addition to K→B and D→B, we show the results for B→K and D→B in Figure 3. In agreement with the results in the main paper, the target error decreases initially, and then increases as more layers are added to the encoder.



(a) W → A (b) D → A

Figure 4. Office-31 dataset. First row: Fixed predictor class, varying number of layers in the encoder. Second row: Fixed total number of layers and optimizing domain-invariant loss in a single intermediate layer.



(a) Vary Layer Number (b) Vary Hidden Size

Figure 5. Effect of predictor complexity on MNIST→MNIST-M. (a) Fix the encoder class and vary the number of layers in the predictor. (b) Fix the encoder class and vary the hidden width of the predictor.

Object Classification In addition to A→W and A→D, we show the results for W→A and D→A in Figure 4. Again, the target error decreases initially and increase as the encoder becomes more complex.

B.3. Effect of Predictor Complexity

Next, we investigate the effect of predictor complexity with the MNIST→MNIST-M data. Following the procedure in Section 5, we augment the original predictor with 1 to 7 additional CNN layers and fix the number of layers in the encoder to 4 or vary the hidden width of the predictor. The results are shown in Figure 5. The target error slightly decreases as the number of layers in the predictor increases, but much less than the 19.8% performance drop when we vary the number of layers in the *encoder* (Section 5.2, see also Fig. 3 and 4): when augmenting the predictor with 7 layers, the target error only decreases by 0.9%. Therefore, we focus on the embedding complexity in the main paper.

C. Computing Ben-David et al. (2010)'s Bound

In Section 7.3 and 7.4, we estimate Ben-David et al. (2010)'s bound with different hypothesis class $\mathcal{H}$. To justify the choice of $\mathcal{H}$, we have to trace back to the proof of Theorem 2 (Ben-David et al., 2010).

Theorem 2. (Ben-David et al., 2010) *For all hypotheses $h \in \mathcal{H}$, the target risk is bounded as*

$$R_T(h) \leq R_S(h) + d_{\mathcal{H}\Delta\mathcal{H}}(p_S, p_T) + \lambda_{\mathcal{H}}, \tag{41}$$

where $\lambda_{\mathcal{H}}$ is the best joint risk

$$\lambda_{\mathcal{H}} \coloneqq \inf_{h' \in \mathcal{H}} [R_S(h') + R_T(h')].$$

Proof. Define the optimal hypothesis h^*:

$$h^* = \arg\inf_{h' \in \mathcal{H}} R_S(h') + R_T(h') \tag{42}$$

We then bound the target risk of h as follows:

$$\begin{aligned}
& R_T(h) && (43)\\
\leq\ & R_T(h, h^*) + R_T(h^*) && (44)\\
\leq\ & |R_S(h, h^*) - R_T(h, h^*)| + R_S(h, h^*) + R_T(h^*) && (45)\\
\leq\ & R_S(h) + |R_S(h, h^*) - R_T(h, h^*)| \\
& \qquad + R_S(h^*) + R_T(h^*) && (46)\\
=\ & R_S(h) + |R_S(h, h^*) - R_T(h, h^*)| + \lambda_{\mathcal{H}} && (47)\\
\leq\ & R_S(h) + \underbrace{\sup_{h, h' \in \mathcal{H}} |R_S(h, h') - R_T(h, h')|}_{\mathcal{H}\Delta\mathcal{H}\text{-Divergence}} + \lambda_{\mathcal{H}} && (48)
\end{aligned}$$

where the first and third inequality follow from the triangle inequality and we replace h^* with h' in the last inequality. □

The tightness of the bound depends on the hypothesis class $\mathcal{H}$ that the $\mathcal{H}\Delta\mathcal{H}$-divergence uses. According to equation (48) in the proof, to take the supremum, we have to ensure that the candidate hypothesis h and the optimal hypothesis h^* belong to $\mathcal{H}$.

For instance, a model h pretrained on the source belongs to $\mathcal{H} = \{h \in \mathcal{H} | \hat{R}_S(h) \leq \epsilon\}$ but not necessarily $\mathcal{P}^{\epsilon}_{\mathcal{FG}}$. However, to estimate the target risk for a domain-invariant classifier $h = fg$ which satisfies $R_S(h) + \alpha d(p^g_S(Z), p^g_T(Z) \leq \epsilon$ (Section 7.4), we can tighten the bound by setting $\mathcal{H} = \mathcal{P}^{\epsilon}_{\mathcal{FG}}$ since $h \in \mathcal{P}^{\epsilon}_{\mathcal{FG}}$. Note that in return, the unobserved best joint risk increases from $\lambda_{\mathcal{H}}$ to $\lambda_{\mathcal{P}^{\epsilon}_{\mathcal{FG}}}$.

We adopt the computational approach described in Section 7.2 to approximate the $\mathcal{H}\Delta\mathcal{H}$-divergence. For instance, to approximate $\mathcal{P}^{\epsilon}_{\mathcal{FG}}\Delta\mathcal{P}^{\epsilon}_{\mathcal{FG}}$-divergence, we optimize the following objective:

$$\begin{aligned}
\max_{fg, f'g' \in \mathcal{FG}} & R_T(fg, f'g') - R_S(fg, f'g') \\
& - \lambda(R_S(fg) + R_S(f'g')) \\
& - \lambda\alpha(d(p^g_S(Z), p^g_T(Z)) + d(p^{g'}_S(Z), p^{g'}_T(Z)))
\end{aligned} \tag{49}$$

where $\lambda > 0$. We empirically estimate the R_T and R_S with the validation set and minimize the objective via standard stochastic gradient descent.

D. Experiment Details and Network Architectures

D.1. Amazon Review Dataset

The learning rate of the Adam optimizer is set to $1 \times e^{-3}$ and the model is trained for 50 epochs. We adopt the original progressive training strategy for the discriminator (Ganin et al., 2016) where the weight α for the domain-invariant loss is initiated at 0 and is gradually changed to 1 using the following schedule:

$$\alpha = \frac{2}{1 + \exp(-10 \cdot p)} - 1, \tag{50}$$

where p is the training progress linearly changing from 0 to 1. The architecture of the hypothesis and discriminator are as follows:

Encoder
nn.Linear(5000, 128)
nn.ReLU
nn.Linear(128, 128)
nn.ReLU
$\times n$ (depends on the number of layers)
Predictor
nn.Linear(128, 128)
nn.ReLU
$\times n$ (depends on the number of layers)
nn.Linear(128, 2)
nn.Softmax

Discriminator
nn.Linear(128, 256)
nn.ReLU
nn.Linear(256, 256)
nn.ReLU
$\times 5$
nn.Linear(256, 2)
nn.Softmax

D.2. Digit Classification

The learning rate of the Adam optimizer is set to $1 \times e^{-3}$ and the model is trained for 100 epochs. The weight α for domain-invariant loss is initiated at 0 and is gradually changed to 0.1 using the same schedule in Section D.1. The tradeoff parameter λ for computing proxy error is set to 50 for all tasks. The architecture of the hypothesis and discriminator are as follows:

Encoder
nn.Conv2d(3, 64, kernel_size=5)
nn.BatchNorm2d
nn.MaxPool2d(2)
nn.ReLU
nn.Conv2d(64, 128, kernel_size=5)
nn.BatchNorm2d
nn.Dropout2d (only added for MNIST→MNIST-M)
nn.MaxPool2d(2)
nn.ReLU
nn.Conv2d(128, 128, kernel_size=3, padding=1)
nn.BatchNorm2d
nn.ReLU
$\times n$ (depends on the number of layers)

Predictor
nn.Conv2d(128, 128, kernel_size=3, padding=1)
nn.BatchNorm2d
nn.ReLU
$\times n$ (depends on the number of layers)
flatten
nn.Linear(2048, 256)
nn.BatchNorm1d
nn.ReLU
nn.Linear(256, 10)
nn.Softmax

Discriminator
nn.Conv2d(128, 256, kernel_size=3, padding=1)
nn.ReLU
nn.Conv2d(256, 256, kernel_size=3, padding=1)
nn.ReLU
$\times 4$
Flatten
nn.Linear(4096, 512)
nn.ReLU
nn.Linear(512, 512)
nn.ReLU
nn.Linear(512, 2)
nn.Softmax

In the hidden width experiments, we use the architectures above as a basis and scale their relevant their hidden widths.

D.3. Office-31

We exploit the ResNet-50 (He et al., 2016) trained on ImageNet (Deng et al., 2009) for feature extraction. The learning rate of the Adam optimizer is set to $3 \times e^{-4}$ and the models are trained for 100 epochs. The weight α for the domain-invariant loss is initiated at 0 and is gradually changed to 1 using the same schedule in Section D.1. The tradeoff parameter λ for computing proxy error is set to 50 for all tasks. The architecture of the hypothesis and discriminator are as follows:

Encoder
nn.Linear(2048, 256)
nn.ReLU
nn.Linear(256, 256)
nn.ReLU
$\times n$ (depends on the number of layers)
Predictor
nn.Linear(256, 256)
nn.ReLU
$\times n$ (depends on the number of layers)
nn.Linear(256, 2)
nn.Softmax

Discriminator
nn.Linear(256, 256)
nn.ReLU
$\times 6$
nn.Linear(256, 2)
nn.Softmax

References

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE*

conference on computer vision and pattern recognition, pp. 770–778, 2016.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.