Semi-Supervised Deep Regression with Uncertainty Consistency and Variational Model Ensembling via Bayesian Neural Networks (Supplementary Materials)

Anonymous submission

Sup-1 : Maximum Likelihood Derivation of Heteroscedastic Regression Loss and Homoscedastic Regression Loss

In this section, we review the maximum likelihood derivation process for interested readers.

We model sample observations using a deterministic component, $f_\mu(x_i|\theta)$, and a stochastic component, ϵ_i such that:

$$y_i = f_\mu(x_i|\theta) + \epsilon_i \ , \ \ \epsilon_i \sim \mathcal{N}(0, \sigma_i^2) \ , \tag{1}$$

where θ represents the parameters of our model. Alternatively, we can write:

$$y_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \ , \ \ \mu_i = f_\mu(x_i|\theta) \tag{2}$$

For maximum likelihood estimation, we aim to find the model parameters θ that maximize the likelihood of us observing our given data. We denote the total likelihood, *i.e.* the total probability of observing the data samples we have with model parameters θ, as $\mathcal{L}(\theta|\mathbf{x}, \mathbf{y})$. Mathematically expressed, we aim to find:

$$\arg\max_\theta \ \mathcal{L}(\theta|\mathbf{x}, \mathbf{y}) = \arg\max_\theta \ \prod_{i=1}^{N} p(y_i|x_i, \theta) \tag{3}$$

This task can be simplified by calculating the negative log likelihood (NLL) value, $-\ln \mathcal{L}(\theta|\mathbf{x}, \mathbf{y})$, and *minimizing* it with respect to θ. For simplicity, we denote NLL as $l(\theta|\mathbf{x}, \mathbf{y})$, which can be expanded to give us:

$$\begin{aligned}
&l(\theta|\mathbf{x}, \mathbf{y}) && (4)\\
&= -\ln \prod_{i=1}^{N} p(y_i|x_i, \theta) && (5)\\
&= -\sum_{i=1}^{N} \ln p(y_i|x_i, \theta) && (6)\\
&= -\sum_{i=1}^{N} \ln \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp \frac{-(y_i - f_\mu(x_i|\theta))^2}{2\sigma_i^2} \right) && (7)\\
&= -\sum_{i=1}^{N} \ln \frac{1}{\sqrt{2\pi\sigma_i^2}} + \sum_{i=1}^{N} \frac{(y_i - f_\mu(x_i|\theta))^2}{2\sigma_i^2} && (8)\\
&= \sum_{i=1}^{N} \frac{\ln 2\pi\sigma_i^2}{2} + \sum_{i=1}^{N} \frac{(y_i - f_\mu(x_i|\theta))^2}{2\sigma_i^2} && (9)
\end{aligned}$$

If we assume homoscedastic errors, *i.e.* σ_i^2 is equal to some constant value for all observations, then the NLL $l(\theta|\mathbf{x}, \mathbf{y})$ is only dependant on the squared error term. Therefore:

$$\arg\min_\theta \ l(\theta|\mathbf{x}, \mathbf{y}) = \arg\min_\theta \sum_{i=1}^{N} (y_i - f_\mu(x_i|\theta))^2 \tag{10}$$

where $f_\mu(x_i|\theta)$ is our predicted value, $\hat{y}$. This gives us our mean squared error loss for standard regression.

Homoscedastic regression tends to be a more restrictive and inaccurate assumption however since most data inherently have observational uncertainty related to the input. For example, facial photographs with more noise may have age labels that are not as accurate. Another example is appearance differences between people aged between 30 and 40 may not be as distinctive as between 1 and 10, which can lead to greater variance in sample observations for ages 30-40.

We can instead model σ_i^2 separately based on the given input data:

$$y_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \ , \ \ \mu_i = f_\mu(x_i|\theta) \ , \ \ \sigma_i^2 = f_{\sigma^2}(x_i|\theta) \tag{11}$$

Minimizing the NLL function then gives us the heteroscedastic regression loss function:

$$\arg\min_{\theta} \; l(\theta|\mathbf{x},\mathbf{y}) \tag{12}$$

$$= \arg\min_{\theta} \left(\sum_{i=1}^{N} \frac{\ln f_{\sigma^2}(x_i|\theta)}{2} + \sum_{i=1}^{N} \frac{(y_i - f_{\mu}(x_i|\theta))^2}{2 f_{\sigma^2}(x_i|\theta)} \right) \tag{13}$$

Sup-2: Reducing Prediction Variance through Ensembling

We show that $E[(\widetilde{y}_i - y_i)^2] \leq E[(\hat{y}_i - y_i)^2]$, *i.e.* the expected performance of the predictor $\widetilde{y}_i$, defined in Eq. 5 is always as good as or better than $\hat{y}_i$. Using bias-variance decomposition, we have:

$$E[(\hat{y}_i - y_i)^2] = \underbrace{(E[\hat{y}_i] - y_i)^2}_{\text{Bias}(\hat{y}_i)} + \underbrace{E[(\hat{y}_i - E[\hat{y}_i])^2]}_{\text{Variance}(\hat{y}_i)}, \tag{13}$$

$$E[(\widetilde{y}_i - y_i)^2] = \underbrace{(E[\widetilde{y}_i] - y_i)^2}_{\text{Bias}(\widetilde{y}_i)} + \underbrace{E[(\widetilde{y}_i - E[\widetilde{y}_i])^2]}_{\text{Variance}(\widetilde{y}_i)}. \tag{14}$$

By substituting $\widetilde{y}_i = \frac{1}{T}\sum_{t=1}^{T} {\hat{y}_i}^t$ into Bias($\widetilde{y}_i$), we have:

$$(E[\widetilde{y}_i] - y_i)^2 = (E[\frac{1}{T}\sum_{t=1}^{T} {\hat{y}_i}^t] - y_i)^2 \tag{15}$$

$$= (\frac{1}{T}\sum_{t=1}^{T} E[{\hat{y}_i}^t] - y_i)^2 \tag{16}$$

$$= (E[\hat{y}_i] - y_i)^2 . \tag{17}$$

Therefore, Bias($\widetilde{y}_i$) = Bias($\hat{y}_i$). For the variance term, Variance($\widetilde{y}_i$):

$$E[(\widetilde{y}_i - E[\widetilde{y}_i])^2] = Var(\widetilde{y}_i) \tag{18}$$

$$= Var(\frac{1}{T}\sum_{t=1}^{T} {\hat{y}_i}^t) \tag{19}$$

$$= \frac{1}{T^2}\sum_{t=1}^{T} Var({\hat{y}_i}^t) \tag{20}$$

$$= \frac{1}{T}\text{Variance}(\hat{y}_i) . \tag{21}$$

Because Bias($\widetilde{y}_i$) = Bias($\hat{y}_i$) and Variance($\widetilde{y}_i$) $\leq$ Variance($\hat{y}_i$), we get:

$$E[(\widetilde{y}_i - y_i)^2] \leq \text{Bias}(\hat{y}_i) + \text{ Variance}(\hat{y}_i) \tag{22}$$

$$E[(\widetilde{y}_i - y_i)^2] \leq E[(\hat{y}_i - y_i)^2] . \tag{23}$$

Therefore, $\widetilde{y}_i$ is a better predictor than $\hat{y}_i$.

Sup-3: Implementation Details for Age Estimation from Photographs

RNDB (Berg, Oskarsson, and O'Connor 2021) We directly follow the implementation and hyper-parameters used by Berg *et al*. (Berg, Oskarsson, and O'Connor 2021)[1], which was also done on the UTKFace dataset. Our results on the fully labeled datasets match their reported results. Experiments for the semi-supervised settings were done using only the labels that are available.

SSDPKL (Jean, Xie, and Ermon 2018) SSDPKL is originally designed to process structured data *i.e.* tabular data inputs, and relies on pretrained feature extractors to obtain one-dimensional embeddings for learning. We pretrain a ResNet-50 encoder for age regression to perform feature extraction, which we find gives better results than using ImageNet pretrained weights. The ResNet-50 model was trained for 30 epochs using learning rate of 1×10^{-4}, weight decay of 10^{-3} using the Adam optimizer, and learning rate decay of 0.1 after 10 epochs. We use a batch size of 32 and MSE loss. Only the available labeled data for each semi-supervised setting was used.

The trained regression model was used to obtain one dimensional feature embeddings of all images by extracting the intermediary features before the fully collected layer. We then follow the implementation by Jean *et al*. (Jean, Xie, and Ermon 2018)[2] and use the feature embeddings as the input. We use latent dimension of 2, learning rate 0.001, and train over 50 iterations, which we find gives the best results.

TNNR (Wetzel, Melko, and Tamblyn 2021) We implement the methodology of TNNR (Wetzel, Melko, and Tamblyn 2021) such that two ResNet-50 encoders with shared weights are used to obtain feature embeddings of a pair of input images. The two feature embeddings are concatenated and a fully connected layer is used to estimate the difference in their labeled values.

Every iteration, we sample a batch of 32 labeled images and train the model to estimate the difference in their label. We also sample a batch of 32 unlabeled images and apply loop consistency. Loops consisting of unlabeled data only, as well as loops consisting of unlabeled and labeled data, are selected. We constrain the model such that the sum of estimated differences between three input pairs forming a loop equal to zero.

The model was trained for 30 epochs using learning rate of 1×10^{-4}, weight decay of 10^{-3} using the Adam optimizer, and learning rate decay of 0.1 after 10 epochs. We set the weight contribution from unlabeled samples to 0.1, which we find gives the best results. During inference, we compare with 5 training samples for which the label is already known and use the estimated difference to calculate the average target prediction.

COREG (Zhou, Li et al. 2005) We modify the COREG implementation by Zhou *et al.* , which originally uses KNN

[1] https://github.com/axeber01/dold

[2] https://github.com/ankurmallick/DPKL

regression on structured tabular inputs, such that it can perform deep image regression. We replace KNN regression with two deep regression models using the ResNet-50 encoder and use predictions on unlabeled data as pseudo-labels for cross supervision. The ResNet-50 model was trained for 30 epochs using learning rate of 1×10^{-4}, weight decay of 10^{-3} using the Adam optimizer, and learning rate decay of 0.1 after 10 epochs. We use a batch size of 32 for both labeled and unlabeled data, and use predictions on unlabeled data as pseudo-labels for each other. We use MSE loss and use a loss weighting of 10 for unlabeled samples, which we find gives the best results.

Baseline The baseline model uses two ResNet-50 BNNs co-trained using heteroscedastic regression loss. We use the same parameters as our proposed methodology, *i.e.* learning rate of 10^{-4}, weight decay of 10^{-3}, and the Adam optimizer. We train for 30 epochs with learning rate decay of 0.1 after 10 epochs. We use a batch size of 32 for sampling labeled and unlabeled data. Pseudo-labels for unlabeled data are calculated as the mean prediction of the two models. We use a weight value of 10 for unlabeled data. Monte-carlo dropout is not used for training or inference. Uncertainty consistency loss and variational model ensembling is not used.

Sup-4: Implementation Details for Ejection Fraction Estimation from Echocardiogram Videos

Ouyang *et al*. (Ouyang et al. 2020) We directly follow the implementation and hyper-parameters used by Ouyang *et al*. (Ouyang et al. 2020) [3], which was performed on the EchoNet-Dynamic dataset. Our results on the fully labeled datasets match their reported results. Experiments for the semi-supervised settings were done using only the labels that are available.

SSDPKL (Jean, Xie, and Ermon 2018) To encode video data into one-dimensional feature embeddings, we use the trained R2+1D ResNet encoder (Tran et al. 2018) from the method by Ouyang *et al*. . The trained encoder was used to obtain one dimensional feature embeddings of all videos by saving the intermediate features before the fully connected regression layer. We then follow the implementation by Jean *et al*. (Jean, Xie, and Ermon 2018)[4] using the feature embedding as input. We use latent dimension of 2, learning rate 0.001, and train over 50 iterations, which gives the best results.

TNNR (Wetzel, Melko, and Tamblyn 2021) We implement the methodology of TNNR (Wetzel, Melko, and Tamblyn 2021) such that two R2+1D ResNet encoders (Tran et al. 2018), pretrained on Kinetics 400 (Kay et al. 2017), with shared weights are used to obtain feature embeddings of a pair of input videos. The two feature embeddings are concatenated and a fully connected layer is used to estimate the difference in their labeled values.

[3]https://github.com/echonet/dynamic

[4]https://github.com/ankurmallick/DPKL

Every iteration, we sample a batch of 10 labeled videos and train the model to estimate the difference in their label. We also sample a batch of 10 unlabeled videos and apply loop consistency, such that the sum of estimated differences between three input pairs forming a loop equal to zero. The model was trained for 25 epochs with learning rate decayed by 0.1 after 15 epochs. We set the weight contribution from unlabeled samples to 0.001, which we find gives the best results. During inference, we compare with 5 training samples for which the label is known and use the estimated difference to calculate the average target prediction.

COREG (Zhou, Li et al. 2005) We modify the COREG implementation by Zhou *et al*. to use two deep regression models based on the R2+1D ResNet encoder (Tran et al. 2018) and use predictions on unlabeled data as pseudo-labels for cross supervision. The R2+1D ResNet model uses pretrained weights from Kinetics 400 (Kay et al. 2017) and was trained for 25 epochs using SGD with learning rate of 1×10^{-4}, momentum of 0.9, and learning rate decay of 0.1 after 15 epochs. We use a batch size of 10 for both labeled and unlabeled data, and use predictions on unlabeled data as pseudo-labels for each other. We use MSE loss and use a loss weighting of 10 for unlabeled samples, which we find gives the best results.

Baseline The baseline model uses two R2+1D ResNet (Tran et al. 2018) BNNs with pretrained weights from Kinetics 400 (Kay et al. 2017) and performs co-training using heteroscedastic regression loss.

The original method used by Ouyang *et al*. (Ouyang et al. 2020) achieved best results without using mean and standard-deviation normalization on the labels, which means the large error terms can lead to unstable heteroscedastic regression without adjustment. We add an additional weighting parameter to the heteroscedastic regression loss function to dampen the effect of the error term:

$$\mathcal{L}_{reg}^{lb} = \frac{1}{N} \sum_{m=a,b} \sum_{i=1}^{N} \left(\frac{(\hat{y}_{i,m} - y_i)^2}{2 \exp(w_{exp}\, \hat{z}_{i,m})} + w_{ali}\, \frac{\hat{z}_{i,m}}{2} \right) . \tag{24}$$

$$\mathcal{L}_{reg}^{ulb} = \frac{1}{N'} \sum_{m=a,b} \sum_{i=1}^{N'} \left(\frac{(\hat{y}_{i,m} - \widetilde{y}_i)^2}{2 \exp(w_{exp}\, \widetilde{z}_i)} + w_{ali}\, \frac{\widetilde{z}_i}{2} \right) . \tag{25}$$

where we set $w_{exp} = 0.05$ and $w_{ali} = 5$.

We use the same parameters as our proposed methodology, *i.e.* learning rate of 10^{-4}, momentum of 0.9, and the SGD optimizer. We train for 25 epochs and decay learning rate by 0.1 after 15 epochs. We use a batch size of 10 for sampling labeled and unlabeled data. Pseudo-labels for unlabeled data are calculated as the mean prediction of the two models. We use a weight value of 10 for unlabeled data. Monte-carlo dropout is not used for training or inference. Uncertainty consistency loss and variational model ensembling is not used.

S-Algorithm 1: Pseudo-code for Training UCVME

Input: D, D'
Output: (f_a, f_b)

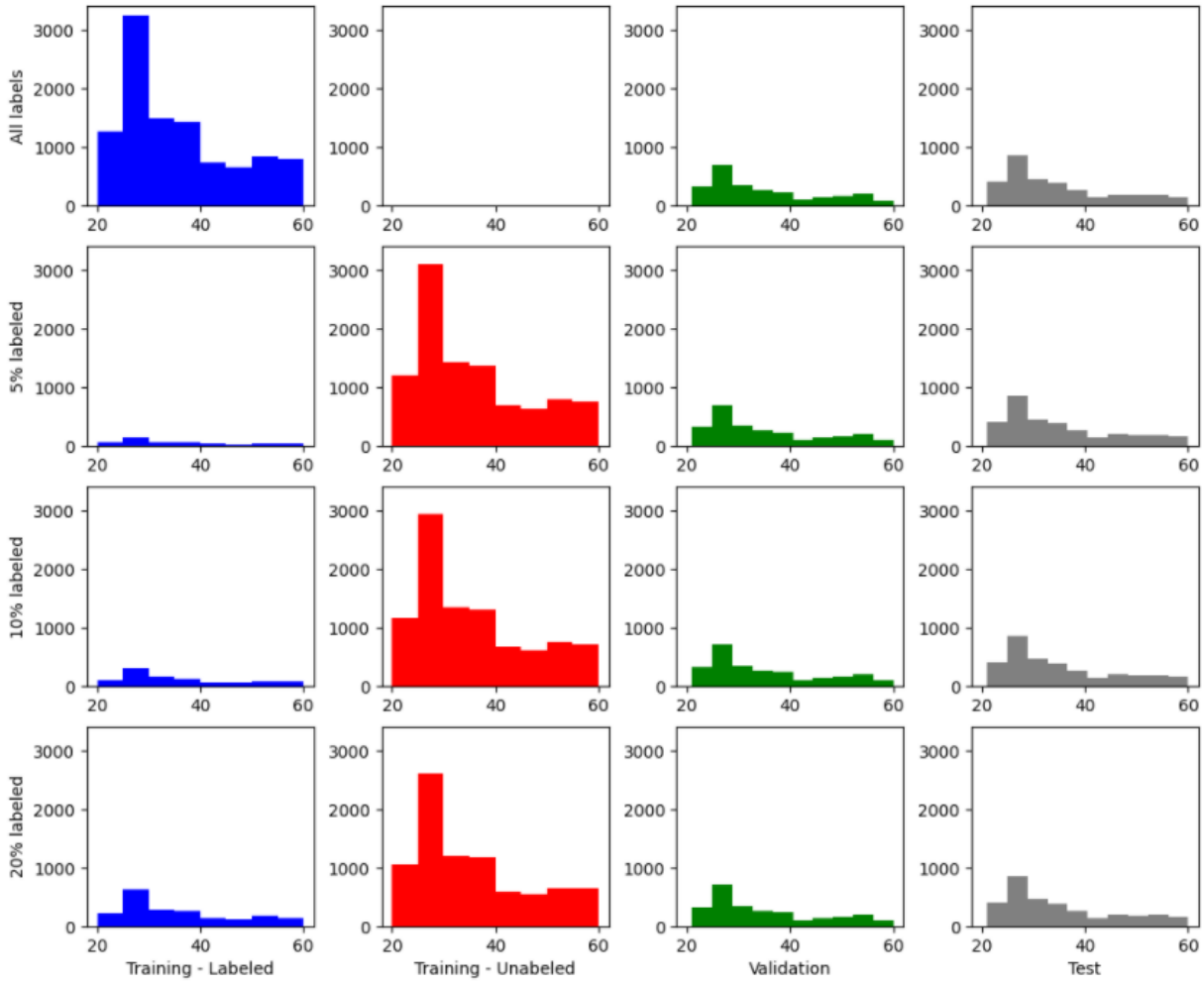
Initialize (f_a, f_b).
while iteration $<$ total iterations **do**
  Sample unlabeled batch $\{x'_{i'}\} \in D'$.
  Calculate pseudo-labels $\{\widetilde{y}'_{i'}, \widetilde{z}'_{i'}\}$ for $\{x'_{i'}\}$ using Eq. 7 and 8.
  Calculate $\mathcal{L}^{ulb}_{reg}$, $\mathcal{L}^{ulb}_{unc}$ for $\{x'_{i'}, \widetilde{y}'_{i'}, \widetilde{z}'_{i'}\}$ using Eq. 9 and 10.
  Sample labeled batch $\{x_i, y_i\} \in D$.
  Calculate $\mathcal{L}^{lb}_{reg}$, $\mathcal{L}^{lb}_{unc}$ for $\{x_i, y_i\}$ using Eq. 2 and 3.
  $\mathcal{L} = \mathcal{L}^{lb}_{reg} + \mathcal{L}^{lb}_{unc} + w_{ulb}\,(\,\mathcal{L}^{ulb}_{reg} + \mathcal{L}^{ulb}_{unc}\,)$
  Update (f_a, f_b) using gradient descent on $\mathcal{L}$
end while
return (f_a, f_b)

S-Table 1: Results for age estimation using different values of w_{ulb}. We show results using only 10% of available training labels. Remaining samples are used as unlabeled data. Results are generally robust for $w_{ulb} < 20$. Best performance is achieved using $w_{ulb} = 10$ and highlighted in **bold**.

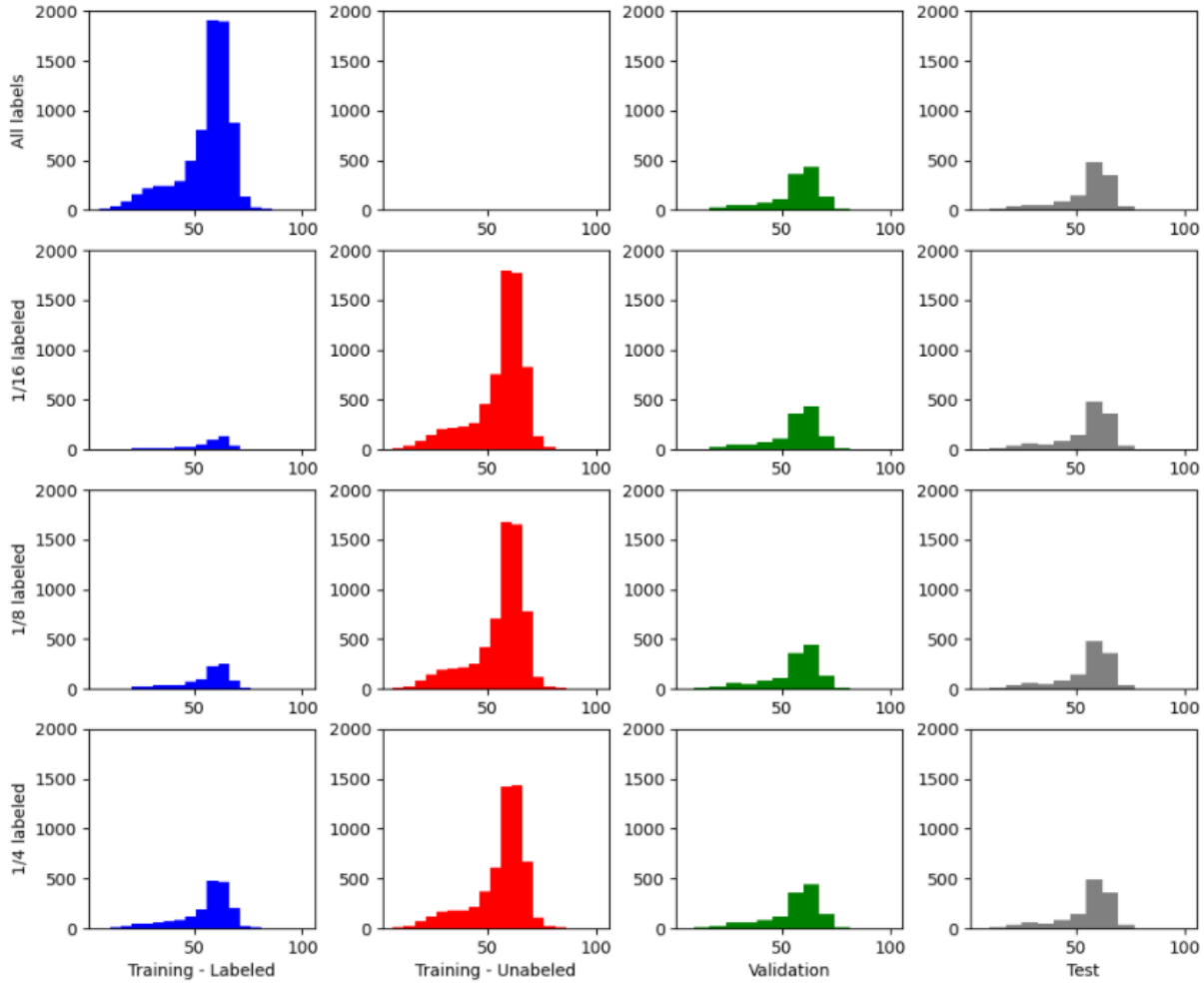
w_{ulb}	**MAE**$\downarrow$	$\mathbf{R}^2 \uparrow$
$w_{ulb} = 5$	5.29 ± 0.04	$57.8\% \pm 0.6$
$w_{ulb} = 10$	$\mathbf{5.26 \pm 0.02}$	$\mathbf{57.9\% \pm 0.3}$
$w_{ulb} = 15$	5.29 ± 0.03	$57.6\% \pm 0.4$
$w_{ulb} = 20$	5.38 ± 0.04	$56.7\% \pm 0.6$

S-Table 2: Results for ejection fraction estimation from echocardiogram videos using different values of w_{ulb}. We show results using only 1/8 of available training labels. Remaining samples are used as unlabeled data. Best performance is achieved using $w_{ulb} = 10$ and highlighted in **bold**.

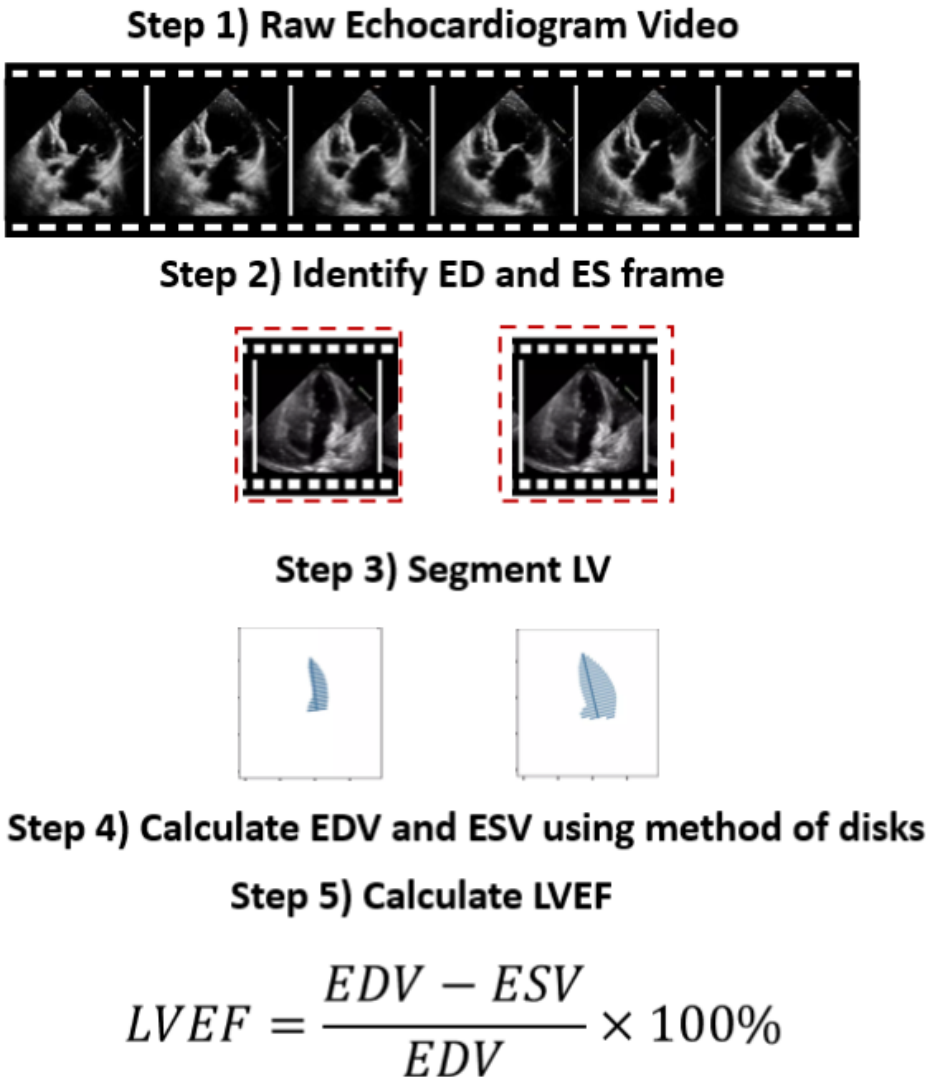
w	**MAE**$\downarrow$	$\mathbf{R}^2 \uparrow$
$w_{ulb} = 5$	5.23 ± 0.02	$66.0\% \pm 0.3$
$w_{ulb} = 10$	$\mathbf{5.10 \pm 0.05}$	$\mathbf{66.6\% \pm 0.5}$
$w_{ulb} = 15$	5.19 ± 0.02	$66.0\% \pm 0.8$
$w_{ulb} = 20$	5.29 ± 0.06	$65.1\% \pm 0.9$



S-Fig. 1: Label distribution of age values for the UTKFace dataset (Zhang, Song, and Qi 2017). We follow training and test splits in previous works by Cao *et al*. (Cao, Mirjalili, and Raschka 2019) and Berg *et al*. (Berg, Oskarsson, and O'Connor 2021). A random subset of the original training split is used for validation. For semi-supervised settings, we randomly split the training set into a labeled and unlabeled dataset.



S-Fig. 2: Label distribution of LVEF values for the EchoNet-Dynamic dataset (Ouyang et al. 2019). We follow the training, validation, and test splits used in the original dataset. For semi-supervised settings, we randomly split the training set into a labeled and unlabeled dataset.



S-Fig. 3: LVEF labeling process for echocardiogram video. End-diastole (ED) and end-systole (ES) frames are first identified from the raw video sequence. These are the phases corresponding to maximum and minimum left ventrical (LV) volume respectively. The LV is then segmented. End-diastole volume (EDV) and end-systole volume (ESV) are estimated from segmentation masks using method of disks (Foley et al. 2012), which approximates 3D volume by using circular disks. Finally, the percentage difference is calculated to find LVEF.

References

Berg, A.; Oskarsson, M.; and O'Connor, M. 2021. Deep ordinal regression with label diversity. In *ICPR*, 2740–2747. IEEE.

Cao, W.; Mirjalili, V.; and Raschka, S. 2019. Rank-consistent ordinal regression for neural networks. *arXiv preprint arXiv:1901.07884*, 1(6): 13.

Foley, T. A.; et al. 2012. Measuring left ventricular ejection fraction-techniques and potential pitfalls. *Eur Cardiol*, 8(2): 108–114.

Jean, N.; Xie, S. M.; and Ermon, S. 2018. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. *NeurIPS*, 31.

Kay, W.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Ouyang, D.; et al. 2019. Echonet-dynamic: a large new cardiac motion video data resource for medical machine learning. In *NeurIPS*.

Ouyang, D.; et al. 2020. Video-based AI for beat-to-beat assessment of cardiac function. *Nature*, 580(7802): 252–256.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 6450–6459.

Wetzel, S. J.; Melko, R. G.; and Tamblyn, I. 2021. Twin Neural Network Regression is a Semi-Supervised Regression Algorithm. *arXiv preprint arXiv:2106.06124*.

Zhang, Z.; Song, Y.; and Qi, H. 2017. Age progression/regression by conditional adversarial autoencoder. In *CVPR*, 5810–5818.

Zhou, Z.-H.; Li, M.; et al. 2005. Semi-supervised regression with co-training. In *IJCAI*, volume 5, 908–913.