Article

A novel interpretable deep transfer learning combining diverse learnable parameters for improved T2D prediction based on single-cell gene regulatory networks

Sumaya Alghamdi and Turki Turki *

Department of Computer Science, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: tturki@kau.edu.sa

Abstract: Accurate deep learning (DL) models to predict type 2 diabetes (T2D) are concerned not only with targeting the discrimination task but also with learning useful feature representation. However, existing DL tools are far from perfect and do not provide appropriate interpretation as a guideline to explain and promote superior performance in the target task. Therefore, we provide an interpretable approach for our presented deep transfer learning (DTL) models to overcome such drawbacks, working as follows. We utilize several pre-trained models including SEResNet152, and SEResNeXt101. Then, we transfer knowledge from pre-trained models via keeping the weights in the convolutional base (i.e., feature extraction part) while modifying the classification part with the use of Adam optimizer to deal with classifying healthy controls and T2D based on single-cell gene regulatory network (SCGRN) images. Another DTL models work in a similar manner but just with keeping weights of the bottom layers in the feature extraction unaltered while updating weights of consecutive layers through training from scratch. Experimental results on the whole 224 SCGRN images using 5-fold cross-validation show that our model (TFeSEResNeXT101) achieving the highest average balanced accuracy (BAC) of 0.97 and thereby significantly outperforming the baseline that resulted in an average BAC of 0.86. Moreover, the simulation study demonstrated that the superiority is attributed to the distributional conformance of model weight parameters obtained with Adam optimizer when coupled with weights from a pre-trained model.

Keywords: deep transfer learning; optimizers; explainable AI; T2D prediction; single-cell gene regulatory network

1. Introduction

Type 2 diabetes (T2D) is a common condition that over time when left untreated can cause damage reaching various organs, including kidney, eye, and heart, to just name a few [1, 2]. Patients with diabetes incur an overall average medical expenditure more than two times that of those without diabetes. Therefore, diabetes is considered as a burden associated with higher medical costs, and increased mortality rates [3]. Obtaining a highly accurate tool to discriminate between healthy and T2D subjects can aid in disease diagnosis, management, prevention and understanding [4]. Therefore, scientific efforts have been made contributing to detect T2D using computational methods [5-8].

Pyrros et al. [9] employed deep learning (DL)-based approach to identify T2D using chest x-ray (CXR) images pertaining to healthy and T2D patients working as follows. They employed the ResNet34 DL model incorporating typical data augmentation with the use of Adam optimizer [10]. The dataset to develop (i.e., train from scratch to induce) the ResNet34 model consisted of 271065 CXR training images in which 45961 were designated as T2D CXR images and the remaining 225104 were as CXR images for healthy control subjects. The trained model was then applied for a testing set consisting of 9943 CXR images. Results demonstrated that the DL model achieved an AUC of 0.84 when compared to an AUC of 0.79 for the baseline linear regression (LR) incorporating only clinical information data. An ensemble of both LR and ResNet34 generated an AUC of 0.85, considered as a marginal improvement in the prediction performance. These results demonstrate the feasibility of DL in screening T2D patients using CXR images. Wachinger et al. [11] presented a DL approach to predict T2D based on neck-to-knee MRI images and clinical information. The MRI images dataset consisted of 3406 MRI images in which the class distribution is uniformed (i.e., 1703 as T2D and 1703 as MRI images for healthy control subjects). The DL approach consists of convolutional layers, maxpooling layers, batch normalization layers, dropout

layer and incorporation of dynamic affine feature map transform (DAFT) within convolutional layers to concatenate features obtained from clinical and MRI image data. Five-fold cross-validation was utilized to assess the performance of the whole data. Results demonstrated the superiority of CNN-DAFT achieving an AUC of 0.871, significantly outperforming CNN using only MRI images and linear regression using only clinical information.

Das [12] et al. presented a learning-based approach combining deep and machine learning for the diagnosis of T2D based on DNA sequences working as follows. First, they transformed the DNA sequence pertaining to healthy and T2D to images, provided as input to ResNet [13] and VGG19 DL models to extract features. Then, providing the extracted features along with corresponding class labels to machine learning algorithms, namely support vector machines(SVM) and KNN. Experimental results using cross-validation on the whole image dataset demonstrate the good performance of SVM when coupled with extracted features using ResNet DL model. Naveed et al. [14] employed DL to predict T2D. The dataset consisted of 19181 patient records data in which 7715 records were for diabetic patients while the remaining 11466 records were for non-diabetic patients. The dataset was divided into training and testing in which training composed of 80% of the dataset while the remaining 20% was assigned for testing. DL models included CNN, LSTM, and CNN-LSTM [15]. Experimental results demonstrated the superior performance of CNN-LSTM achieving the highest performance results when compared to other models including decision tree and SVM. Specifically, CNN-LSTM generated an accuracy of 91.6, and F1-Score of 89.2. These results demonstrate the feasibility of DL in early predicting T2D. Other AI-driven computational methods have been proposed to aid in predicting T2D [8, 16].

As inferred single-cell gene regulatory networks (SCGRNs) encode the molecular interactions pertaining to components of specific cell types and thereby can aid in characterizing cellular differentiation in healthy and disease subjects [17, 18], Turki et al. [19] presented a novel DL approach to discriminate between heathy controls and T2D based on SCGRN images working as follows. Because rapid progress in single-cell technologies has contributed to the availability of biological experiments pertaining to gene regulatory networks (GRNs), single-cell gene expression data from the ArrayExpress repository was processed with the use of bigSCale and NetBioV packages [20-22], generating 224 SCGRN images. The class distribution was distributed evenly in terms of healthy controls and T2D images. Then, utilizing RMSprop optimizer [15] with the following DL models: VGG16 [23], VGG19 [23], Xception [24], ResNet50 [13], ResNet101 [13], DenseNet121 [25], and DenseNet169 [25] to discriminate between healthy controls and T2D SCGRN images. Experimental results demonstrated the VGG19 performed better than studied DL models. However, no interpretation was provided to back the prediction performance.

Although these recently developed methods aimed to address the task of prediction T2D, these methods are still far from perfect and do not provide interpretation for practical deep transfer learning (DTL) models aiding in the explanation of the performance superiority. Therefore, this study is unique in the following aspects: (1) we present highly accurate DTL models working by combining weight parameters from pre-trained models and weights obtained with the use of Adam optimizer; (2) we provide, to the best of our knowledge, the first interpretation behind DTL models inspecting and quantifying that conformance of pre-trained model weight parameters with weight parameters obtained with the incorporation of Adam and RMSprop Optimizers. This interpretation framework can guide in the process of designing highly efficient DTL models applicable to wide range of problems; and (3) we conduct experimental study to report the prediction performance and computational running time for the task of predicting single-cell gene regulatory network images pertaining to healthy controls and T2D. Experimental results demonstrate the superiority of our DTL model, TFeSEResNeXT101, performing better than the baseline with 11% improvements. In terms of the running time, our DL models exhibited a significant reduction in training time attributed to transfer learning, which reduced the number of trainable weight parameters. In addition, simulation study unveiled the conformance of parameter weights of both transfer weights from pre-trained models with weights obtained from Adam optimizer as compared to RMSprop that was used by the baseline and resulted in inferior prediction performance, attributed to the divergence of its weight parameters from the weight parameters of pre-trained models.

2. Materials and Methods

2.1. Biological Networks

We provide an illustration in Figure 1 for the biological network images used in this study, which were downloaded from [19] and consisted of 224 SCGRN images pertaining to healthy and type 2 diabetes. The class distribution for these biological network images is balanced (i.e., 224 divided evenly into the two classes). These biological network images were produced with the help of bigSCale package to process the single-cell gene expression data and build regulatory networks,

then visualizing networks via the NetBioV package. In terms of the single-cell gene expression data pertaining to healthy controls and T2D patients, it was obtained from ArrayExpress repository under accession number E-MTAB-5061 [26].

2.2. Deep Transfer Learning

Figure 1 demonstrates how our deep transfer learning (DTL) approach is performed. First, we adapt the following pre-trained models: VGG19, DenseNet201 [25], InceptionV3 [27], ResNet50V2 [28], ResNet101V2 [28], SEResNet152 [29], and SEResNeXt101 [29]. Each pre-trained model has a feature extraction part (i.e., series of convolutional and pooling layers) for feature extraction and a densely connected classifier for classification. Then, we keep the weights unchanged for the feature extraction part of a pre-trained model and change the densely connected classifier to deal with binary classification instead of 1000 classes. Therefore, when feeding the SCGRN image dataset, we extract features using weights of pre-trained models while training the densely connected classifier from scratch and performing prediction. We refer to models using this type of DTL computations as TFeVGG19, TFeDenseNet201, TFeInceptionV3, TFeResNet50V2, TFeResNet101V2, TFeSEResNet152, and TFeSEResNeXT101 (see Figure 1). For the other DTL computations, we keep weights of the bottom layers unchanged in the feature extraction part while performing training from scratch to change weights of top layers in feature extraction part and densely connected layers.

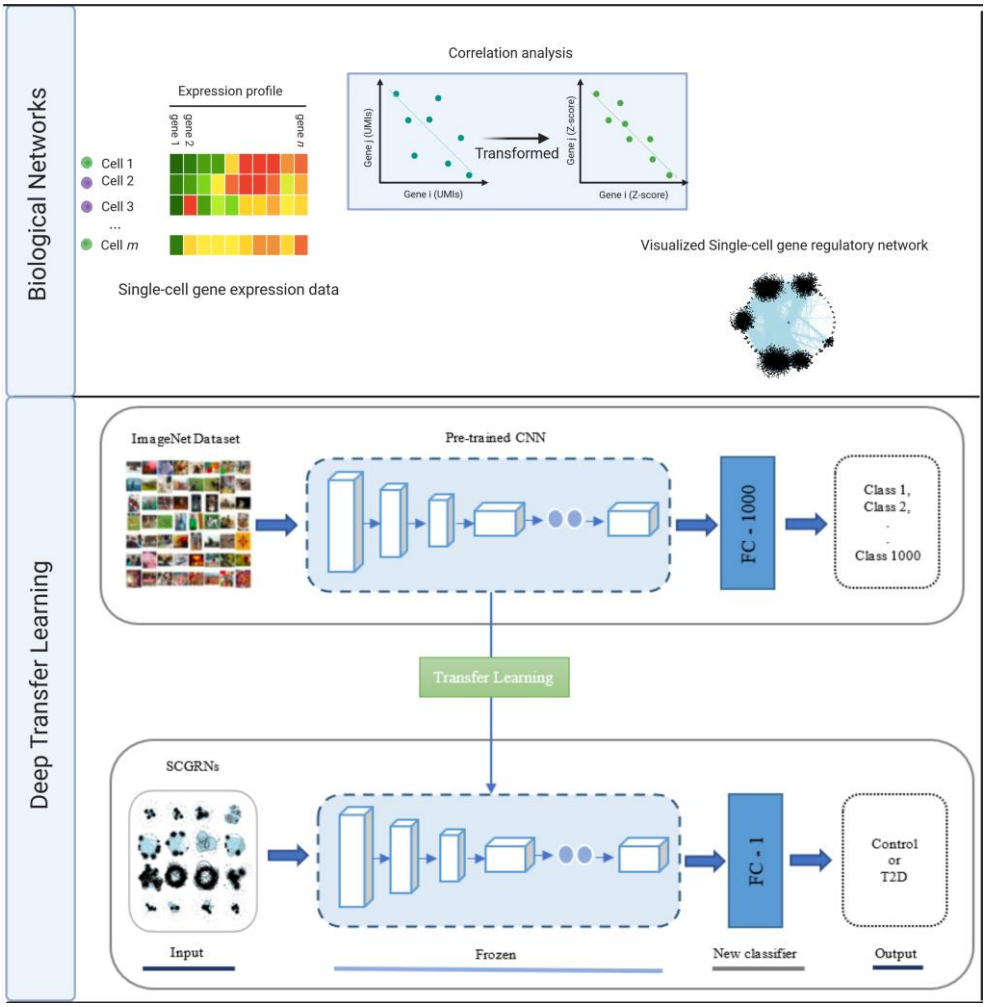

Figure 1: Flowchart of the deep transfer learning-based approach for the predicting T2D using SCGRNs. Biological Networks: To infer single-cell gene regulatory network (SCGRN), gene expression data are provided to bigSCale (performing clustering and differential expression analysis) changing measured correlation between genes from expression values to Z-score, followed by retaining significant correlations to guide in building a regulatory network . A visualization is performed using NetBioV. Deep Transfer Learning: Transfer learning applying feature extraction with new classifier (TFe) to distinguish between T2D and healthy control SCGRNs.

As in TFe-based models, we modify the densely connected classifier dealing with binary classification problem before performing the training phase. As seen in Figure 2, we refer to models employing this type of deep transfer learning as TFtVGG19, TFtDenseNet201, TFtInceptionV3, TFtResNet50V2, TFtResNet101V2, TFtSEResNet152, and TFtSEResNeXT101.

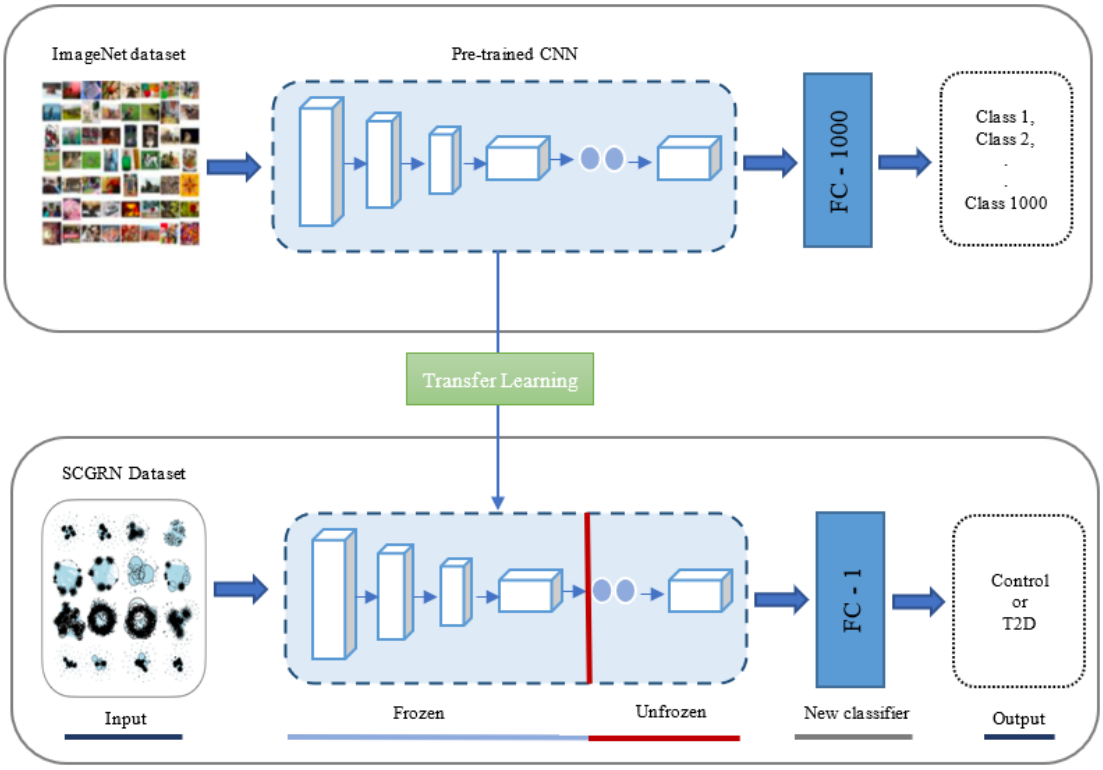

Figure 2: Transfer learning applying fine tuning with new classifier (TFt) to distinguish between T2D and healthy control SCGRNs.

When changing weights during training, we employed three optimizers: Adam, RMSprop, and SGD [30]. When weights are kept unchanged referring to the transfer of knowledge from pre-trained models using SGD optimizer. In terms of predictions of unseen SCGRN images, predictions are mapped to healthy control subjects if the predicted values are greater than 0.5. Otherwise, predictions are mapped to T2D.

3. Results

3.1. Classification Methodology

In this study, we considered seven pre-trained models, namely VGG19, DenseNet201, InceptionV3, ResNet50V2, ResNet101V2, SEResNet152, and SEResNeXt10. Each of the pre-trained models was trained on 1.28 million images from ImageNet database to classify images into 1000 different categories. In terms of TFe-based models, we used the feature extraction part of pre-trained models in which weights were kept unchanged and were used to extract feature from SCGRN images. Moreover, the densely connected classifier was trained from scratch to handle the binary class classification problem. Regarding the TFt-based models, we trained the top layers and densely connected classifier from scratch while retaining the weights of bottom layers unchanged in the feature extraction part. For both TFt-based and TFe-based models, we employed Adam optimizer when updating weights of layers. Moreover, we compared the performance of our deep transfer learning approaches using different optimizers including the baseline (i.e., RMSprop optimizer) as well as against training models from scratch. We set optimization parameters as follows: 0.00001 for the learning rate, 10 for the number of epochs, and 32 for the batch size. In terms of the loss function, we utilized categorical cross-entropy [31].

To assess the performance of studied models, we employed Balanced Accuracy (BAC), Accuracy (ACC), Precision (PRE), Recall (REC), and F1 computed as follows:

$$\mathrm{BAC} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{1}$$

$$\mathrm{ACC} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

$$\mathrm{PRE} = \frac{TP}{TP+FP} \tag{3}$$

$$\mathrm{REC} = \frac{TP}{TP+FN} \tag{4}$$

$$\mathrm{F1} = \frac{2 * PRE * REC}{PRE + REC} \tag{5}$$

where *TN* designates true negative, corresponding to the number of T2D images that were correctly predicted as T2D. *FP* designates false positive, corresponding to the number of T2D images that were incorrectly predicted as healthy controls. *TP* designates true positive, corresponding to the number of healthy control images that were correctly predicted as healthy controls. *FN* designates false negative, corresponding to the number of healthy control images that were incorrectly predicted as T2D.

To evaluate the results on the whole SCGRN image dataset, we employed 5-fold cross-validation as follows. We partitioned the SCGRN image datasets and randomly assigned images into 5 folds. During the first run of 5-fold cross-validation, we used 4 of the folds to train our deep learning models and perform predictions to the remaining fold for testing and record the performance results. Such a process was repeated for an additional 4 runs in which performance results were recorded. Finally, we report the average performance results corresponding to the results obtained from 5-fold cross-validation.

3.2. Implementation Details

All experiments were run on a machine equipped with central processing unit (CPU) of Google Colab. The specifications of CPU runtime offered by Google Colab were Intel Xeon Processor with two cores with 2.30 GHz and 13 GB RAM where the installed version of Python is 3.10.11. For the analysis of models, we used R statistical software [32] to run the experiments and utilized the optimg package in R to run Adam optimizer [33]. All plots were performed using Matplotlib package in python [34].

3.3. Classification Results

3.3.1. Training Results

In Figure 3, we illustrate the training accuracy performance results when running 5-fold cross-validation. It can be seen that our models outperformed all other models trained from scratch. Specifically, TFeVGG19 and TFtVGG19 achieved average accuracies of 0.976 and 0.962, respectively, while VGG19 achieved an average accuracy of 0.530. TFeDenseNet201 outperformed DensNet201 via achieving an average accuracy of 0.988 while DenseNet201 performed better than TFtDenseNet201 via achieving an average accuracy of 0.982 compared to 0.946. For TFe- and TFt-based models when coupled with ResNet101V2, SEResNet152 and SEResNetXT101, they outperformed their counterparts when not applying deep transfer learning (DTL) models. These superior performance results are attributed to the learned representation using transfer learning.

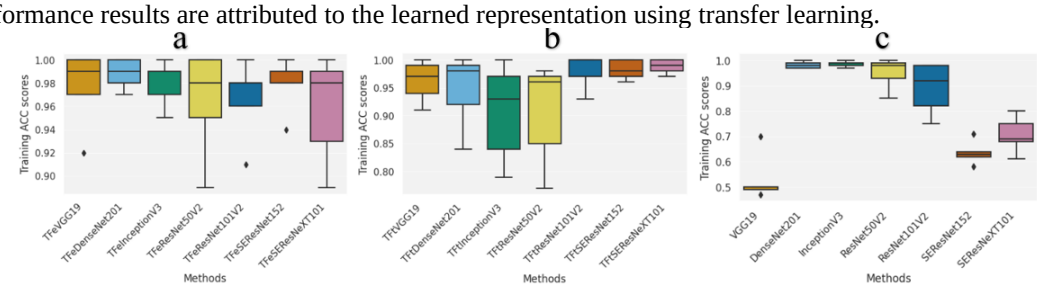

Figure 3: The boxplots presenting the average 5-fold cross-validation results using the ACC measure for the training folds. (a) Deep transfer learning models using feature extraction (referred with the prefix TFe). (b) Deep transfer learning models using fine tuning (referred with the prefix TFt). (c) Deep learning models trained from scratch. ACC is accuracy.

3.3.2. Testing Results

Figures 4 and 5 report the generalization (i.e., test) accuracy performance results and combined confusion matrices, respectively, when 5-fold cross-validation is utilized. TFeSeResNeXT101 achieved the highest average accuracy of 0.968.

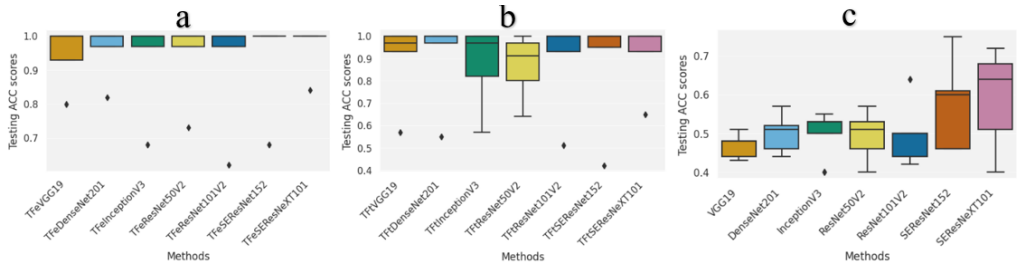

Figure 4: The boxplots presenting the average 5-fold cross-validation results using the ACC measure for the testing folds. (a) Deep transfer learning models using feature extraction (referred with the prefix TFe). (b) Deep transfer learning models using fine tuning (referred with the prefix TFt). (c) Deep learning models trained from scratch. Acc is accuracy.

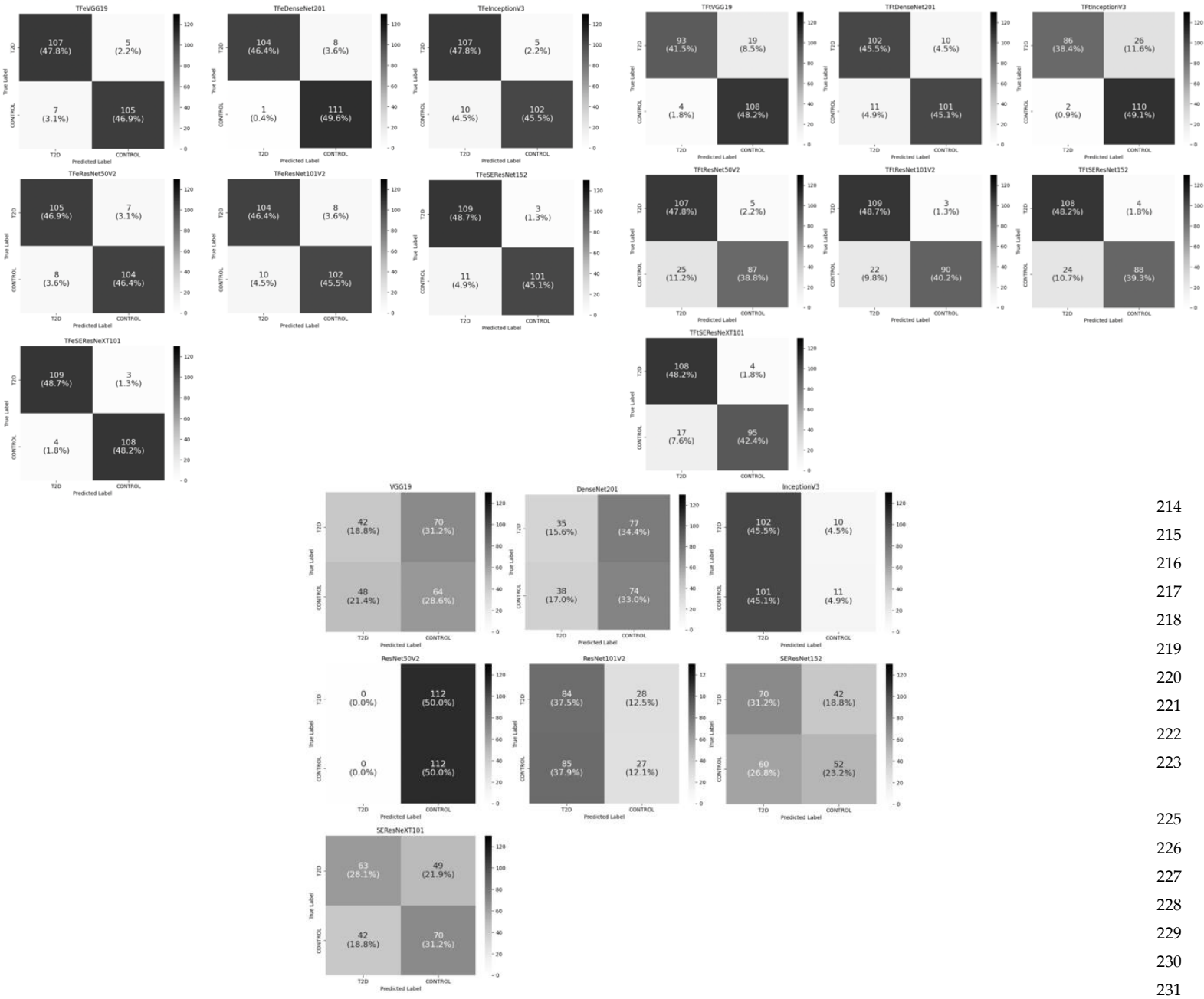

Figure 5: Combined confusion matrices for all methods during the running of 5-fold cross-validation.

The second-best model is TFeDenseNet201, achieving an average accuracy of 0.958, followed by TFeVGG19, TFeResNet50V2, TFeSEResNet152, TFeInceptionV3, and TFeResNet101V2 (generating average accuracies of 0.946, 0.940, 0.936, 0.930, and 0.918, respectively). TFt-based models also outperformed all models trained from scratch (see Figure 4(b and c)). Particularly, TFt-based models generated average accuracies lower and upper bounded by 0.864 and 0.916, respectively, while models trained from scratch were lower and upper bounded by average accuracies of 0.468 and 0.590. These results demonstrate the superior performance of models employing our DTL computations.

In terms of reporting testing performance results using different metrics, our model TFeSeResneXT101 outperforms all other models (see Table 1) via achieving an average BAC of 0.97, average PRE of 0.97 (tie with our model TFeSEResNet152), and average F1 of 0.97. Moreover, TFeVGG19 and TFtVGG19 perform better than VGG19. Similarly, TFeDenseNet201, TFeInceptionV3, TFeResNet50V2, TFeResNet101V2, and TFeSEResNet152 performed better than DenseNet201, InceptionV3, ResNet50V2, ResNet101V2, and SEResNet152, respectively. Th same holds true for TFt-based models outperforming their counterparts (i.e., VGG19, DenseNet201, InceptionV3, ResNet50V2, ResNet101V2, SEResNet152, and SEResNetXT101)

Table 1: Reported average performance results during the running of 5-fold cross-validation on testing using studied models. BAC is balanced accuracy. PRE is precision. REC is recall. The best overall result is underlined and is shown in bold. The method outperforming its counterparts is just underlined.

Model	Optimizer	BAC	PRE	REC	F1
TFeVGG19	Adam	0.94	0.95	0.94	0.94
TFtVGG19	Adam	0.90	0.82	**0.97**	0.89
VGG19	Adam	0.50	0.37	0.46	0.41
TFeDenseNet201	Adam	0.96	0.95	0.95	0.96
TFtDenseNet201	Adam	0.90	0.91	0.90	0.90
DenseNet201	Adam	0.50	0.20	0.51	0.29
TFeInceptionV3	Adam	0.93	0.95	0.91	0.92
TFtInceptionV3	Adam	0.87	0.76	0.92	0.83
InceptionV3	Adam	0.50	0.91	0.50	0.64
TFeResNet50V2	Adam	0.93	0.93	0.92	0.93
TFtResNet50V2	Adam	0.87	0.95	0.81	0.87
ResNet50V2	Adam	0.50	-	-	-
TFeResNet101V2	Adam	0.92	0.92	0.91	0.92
TFtResNet101V2	Adam	0.89	0.90	0.86	0.88
ResNet101V2	Adam	0.52	0.75	0.49	0.59
TFeSEResNet152	Adam	0.94	**0.97**	0.90	0.93
TFtSEResNet152	Adam	0.88	0.96	0.81	0.88
SEResNet152	Adam	0.55	0.67	0.50	0.57
TFeSEResNeXT101	Adam	**0.97**	**0.97**	0.96	**0.97**
TFtSEResNeXT101	Adam	0.90	0.96	0.90	0.91
SEResNeXT101	Adam	0.60	0.56	0.60	0.58

Table 2 reports our best DTL models with different optimizers. It can be shown that TFeSeResNeXT101 and TFtSeResNeXT101 generate the highest performance results when coupled with Adam optimizer method. Specifically, TFeSEResNeXT101 with Adam optimizer generates the highest average BAC (and F1) of 0.97 (and 0.97). TFtSEResNeXT101 with Adam optimizer archives the highest average BAC of 0.91, highest average F1 of 0.90 (tie with SGD optimizer). When TFeSEResNeXT101 and TFtSEResNeXT101 are coupled with SGD optimizer, they generate inferior performance results.

Table 2: Performance comparison of our best deep transfer learning model under different optimizers during the 5-fold cross-validation. BAC is balanced accuracy. Best performance result is shown in bold.

Model	Optimizer	BAC	F1
TFeSEResNeXT101	Adam	**0.97**	**0.97**
	RMSprop	0.92	0.92
	SGD	0.77	0.77
TFtSEResNeXT101	Adam	**0.91**	**0.90**
	RMSprop	0.90	0.89
	SGD	0.90	**0.90**

In Table 3, we compare our model TFeVGG19 with Adam optimizer against the best performing baseline TFeVGG19 with RMSprop optimizer, named VGG19 in [19]. It is evident that our model TFeVGG19 with Adam optimizer achieves the highest average BAC of 0.94 while the baseline obtained an average BAC of 0.86. Moreover, when F1 performance measure is considered, TFeVGG19 with Adam optimizer attains the highest average F1 of 0.94 while the baseline achieved an average F1 of 0.88. The same holds true for TFtVGG19, which achieved the highest average BAC of 0.91, highest average F1 of 0.90.

Table 3: Performance comparison of our deep transfer learning model against recent baseline methods when 5-fold cross-validation is employed. BAC is balanced accuracy. Best performance result is shown in bold.

Model	Optimizer	BAC	F1
TFeVGG19	Adam (Ours)	**0.94**	**0.94**
	RMSprop (Baseline)	0.86	0.88
TFtVGG19	Adam (Ours)	**0.91**	**0.90**
	RMSprop (Baseline)	0.89	0.89

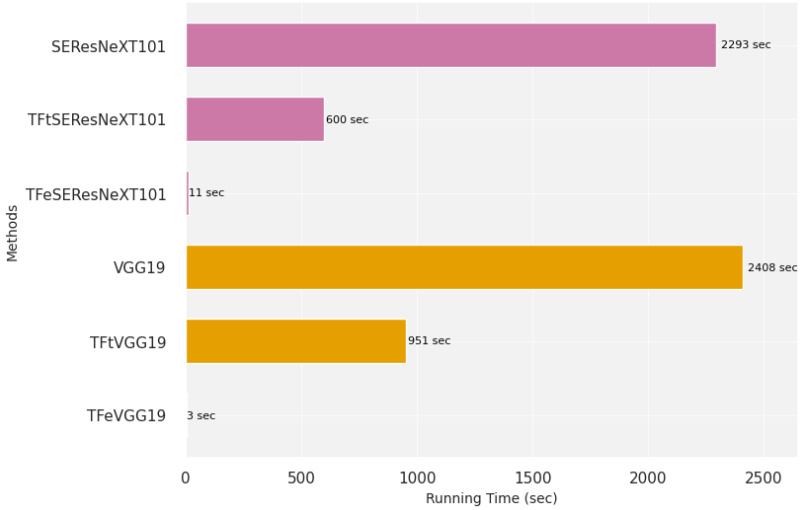

Figure 6: Running time comparisons in seconds for selected models when running 5-fold cross-validation.

In Figure 6, we report the running time in seconds for the process of running 5-fold cross-validation when utilizing our best model (TFeSEResNeXT101) and TFtSEResNeXT101 compared to their peer SeResneXT101. Our model TFeSEResNeXT101 is 208.45x faster than SEResNeXT101. Also, our model TFtSEResNeXT101 is 3.82x faster than SEResNeXT101. Moreover, TFeVGG19 and TFtVGG19 are 802.67x and 2.53x, respectively, faster than VGG19. These

results demonstrate the computational efficiency of the DTL models, in addition to the highly achieved performance results.

3.4. Models Introspection

3.4.1. Stochastic Gradient Descent (SGD)

To minimize the objective function $Q(\theta_0, \theta_1)$ for parameters θ_0 and θ_1 of model $H(x_i)$, we employ gradient descent optimization algorithms to find θ_0 and θ_1 minimizing the objective function. The optimization problem can be formulated as follows:

$$\begin{aligned} \min_{\theta_0,\theta_1} Q(\theta_0, \theta_1) &= \frac{1}{m}\sum_{i=1}^{m} Q_i(\theta_0, \theta_1) \\ &= \frac{1}{m}\sum_{i=1}^{m} (H(x_i) - y_i)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2 \end{aligned} \tag{6}$$

We utilize SGD, RMSprop, and Adam optimization algorithm to minimize the objective function and estimate the model parameters. For SGD, we initialize the parameters θ_0 and θ_1 according to the uniform distribution $U(0,1)$ and setting the learning rate $\eta = 0.001$, maximum number of iterations to 3000. Then, in each time, shuffling the data of *m* examples followed by looping *m* times over the following to update model parameters After the end of looping, the algorithm stops if the maximum number of iterations is reached or $\|\nabla Q(\theta_0, \theta_1)\| \leq 0.001$:

$$\begin{aligned} \theta_0 &:= \theta_0 - \eta.\frac{\partial Q_i}{\partial \theta_0} \\ &:= \theta_0 - \eta.2(\theta_0 + \theta_1 x_i - y_i) \\ \theta_1 &:= \theta_1 - \eta.\frac{\partial Q_i}{\partial \theta_1} \\ &:= \theta_1 - \eta.2x_i(\theta_0 + \theta_1 x_i - y_i) \end{aligned} \tag{7}$$

3.4.2. Root Mean Square Propagation (RMSprop)

For RMSprop, we initialize model parameters according to uniform distribution $U(0,1)$, set the learning rate $\eta = 0.001$, maximum number of iterations to 3000, $\beta = 0.9$, $\epsilon = 10^{-6}$, BatchSize = 16, referred in the following as $|S|$. Then, looping to update model parameters according to each selected batch. After ending of looping over all selected batches, the algorithm terminates when $\|\nabla Q(\theta_0, \theta_1)\| \leq 0.001$ or the maximum number of iterations is reached:

$$
\begin{aligned}
v_{\theta_0} &:= \beta.v_{\theta_0} + (1-\beta)(\tfrac{\partial Q}{\partial \theta_0})^2 \\
&:= \beta.v_{\theta_0} + (1-\beta)(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i))^2 \\
v_{\theta_1} &:= \beta.v_{\theta_1} + (1-\beta)(\tfrac{\partial Q}{\partial \theta_1})^2 \\
&:= \beta.v_{\theta_1} + (1-\beta)(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i).x_i)^2 \\
\theta_0 &:= \theta_0 - \tfrac{\eta}{\sqrt{v_{\theta_0}+\epsilon}}\tfrac{\partial Q}{\partial \theta_0} \\
&:= \theta_0 - \tfrac{\eta}{\sqrt{v_{\theta_0}+\epsilon}}(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i)) \\
\theta_1 &:= \theta_1 - \tfrac{\eta}{\sqrt{v_{\theta_1}+\epsilon}}\tfrac{\partial Q}{\partial \theta_1} \\
&:= \theta_1 - \tfrac{\eta}{\sqrt{v_{\theta_1}+\epsilon}}(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i)x_i)
\end{aligned}
\tag{8}
$$

3.4.3. Adaptive Moment Estimation (Adam)

In terms of Adam, $m_{\theta_0}, m_{\theta_1}, v_{\theta_0}, v_{\theta_1}, \beta_1, \beta_2, \epsilon, and\ \eta$ are initialized as in [35]. Then, looping to update model parameters according to all selected batches of examples. After ending of looping over all selected batches, the algorithm stops if the maximum number of iterations (i.e., 3000) is reached or $\|\nabla Q(\theta_0, \theta_1)\| \leq 0.001$:

$$
\begin{aligned}
m_{\theta_0} &:= \beta_1.m_{\theta_0} + (1-\beta_1)(\tfrac{\partial Q}{\partial \theta_0}) \\
&:= \beta_1.m_{\theta_0} + (1-\beta_1)(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i)) \\
m_{\theta_1} &:= \beta_1.m_{\theta_1} + (1-\beta_1)(\tfrac{\partial Q}{\partial \theta_1}) \\
&:= \beta_1.m_{\theta_1} + (1-\beta_1)(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i)x_i) \\
v_{\theta_0} &:= \beta_2.v_{\theta_0} + (1-\beta_2)(\tfrac{\partial Q}{\partial \theta_0})^2 \\
&:= \beta_2.v_{\theta_0} + (1-\beta_2)(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i))^2 \\
v_{\theta_1} &:= \beta_2.v_{\theta_1} + (1-\beta_2)(\tfrac{\partial Q}{\partial \theta_1})^2 \\
&:= \beta_2.v_{\theta_1} + (1-\beta_2)(\tfrac{2}{|S|}\textstyle\sum_i(\theta_0 + \theta_1 x_i - y_i).x_i)^2 \\
\widehat{m}_{\theta_0} &:= \tfrac{m_{\theta_0}}{(1-\beta_1)} \\
\widehat{m}_{\theta_1} &:= \tfrac{m_{\theta_1}}{(1-\beta_1)} \\
\widehat{v}_{\theta_0} &:= \tfrac{v_{\theta_0}}{(1-\beta_2)} \\
\widehat{v}_{\theta_1} &:= \tfrac{v_{\theta_1}}{(1-\beta_2)} \\
\theta_0 &:= \theta_0 - \tfrac{\eta}{\sqrt{\widehat{v}_{\theta_0}}+\epsilon}\widehat{m}_{\theta_0} \\
\theta_1 &:= \theta_1 - \tfrac{\eta}{\sqrt{\widehat{v}_{\theta_1}}+\epsilon}\widehat{m}_{\theta_1}
\end{aligned}
\tag{9}
$$

3.4.4. Simulated Data

To demonstrate the efficiency of the proposed deep transfer learning (DTL) models incorporation mixed parameters derived from both SGD and Adam, we conducted simulation studies to explain the superiority behind the proposed models as well as imitate the numerical behavior. Particularly, we consider the following four predictive models:

$$\mathrm{F}_1(x) = 0.5 + 0.79x + \epsilon \tag{10}$$

$$\mathrm{F}_2(x) = 1 - \exp(\tfrac{-1}{2x}) \tag{11}$$

$$\mathrm{F}_3(x) = 0.7 + 3x^2 + \epsilon \tag{12}$$

$$\mathrm{F}_4(x) = (6x - 2)^2 + \sin(12x - 4) \tag{13}$$

where $X \sim U(0,1)$ and $\epsilon \sim N(0,0.2)$ in which $U()$ and $N()$ are uniform and normal distributions, respectively. For F_1, when we have X and Y=$\mathrm{F}_1(X)$, we perform the following steps. Let $H_{SGD}(x_i) = \theta_0 + \theta_1 x_i$ (for i =1..m) be the model in which we want to estimate parameters using (X,Y) data from F_1 coupled with Equation 7. Similarly, let $H_{RMSprop}(x_i)$ *and* $H_{Adam}(x_i)$ be models in which we want to estimate their parameters using Equations 8 and 9, respectively, coupled with (X,Y) data from F_1. Then, we provide each $x_i \in X$ to perform predictions corresponding to y_i'. In Figure 7, we report 2D plots for X and predicted $Y' = \{y_1', \dots, y_m'\}$ via each model using data generated according to Equation 10, where SGD refers to plotting $(x_i, H_{SGD}(x_i))$ while Adam and RMSprop refers to plotting results obtained via $(x_i, H_{Adam}(x_i))$ and $(x_i, H_{RMSprop}(x_i))$, respectively, and i=1..m. We then repeat this process for an additional 8 runs. Therefore, we have 9 runs in total

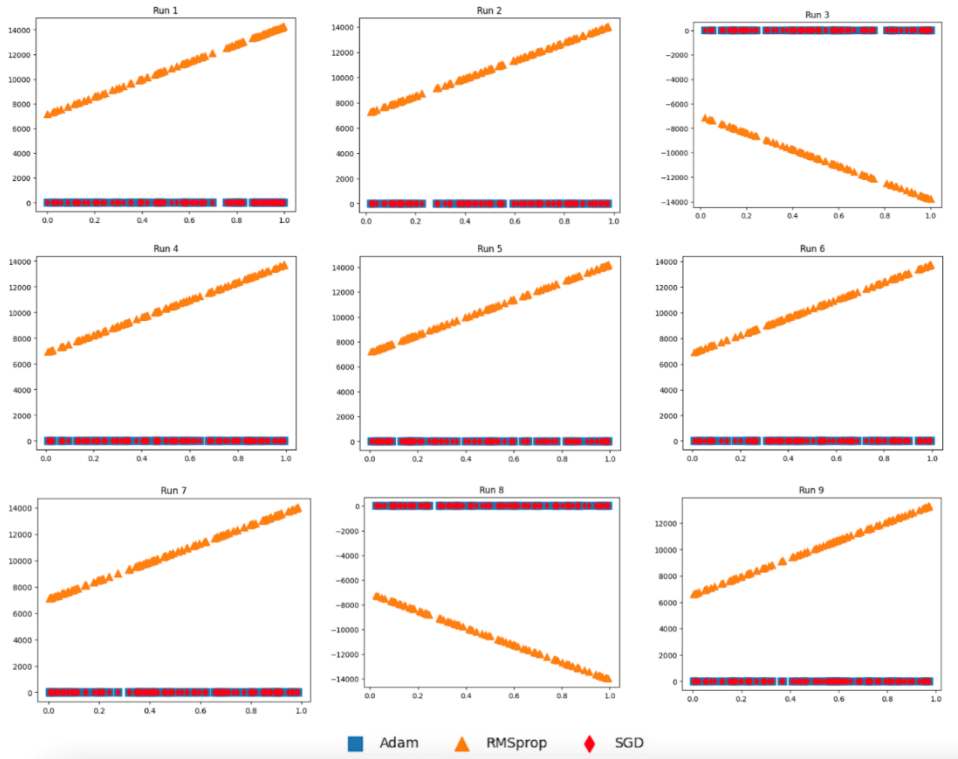

Figure 7: Plots for the three models as $(x_i, H_{SGD}(x_i))$, $(x_i, H_{Adam}(x_i))$, and $(x_i, H_{RMSprop}(x_i))$ for i=1..m according to x_i generated using F_1.

It can be seen from Figure 7 that model induced via RMSprop has more distributional differences compared to those obtained via SGD and Adam. To quantify distributional differences between SGD and Adam against SGD and RMSprop, we perform the following computations:

$$d_{SA} = \sqrt{\sum_i \left(\left(x_i^{SGD} - x_i^{Adam} \right)^2 + \left(z_i^{SGD} - z_i^{Adam} \right)^2 \right)} \tag{14}$$

$$d_{SR} = \sqrt{\sum_i \left(\left(x_i^{SGD} - x_i^{RMSprop} \right)^2 + \left(z_i^{SGD} - z_i^{RMSprop} \right)^2 \right)} \tag{15}$$

where d_{SA} measuring the distance between data associated with SGD and Adam. Similarly, d_{SR} measures the distance between data associated with SGD and RMSprop. The lower the distance value, the less the distribution difference is. Figure 11(a) plots d_{SA} and d_{SR} for the 9 runs. It can be seen that $H_{Adam}(x_i)$ is closer to $H_{SGD}(x_i)$ than $H_{RMSprop}(x_i)$ to $H_{SGD}(x_i)$ in most runs. Moreover, the distributional differences are statistically significant (P-value = 7.28×10^{-14} from t-test).

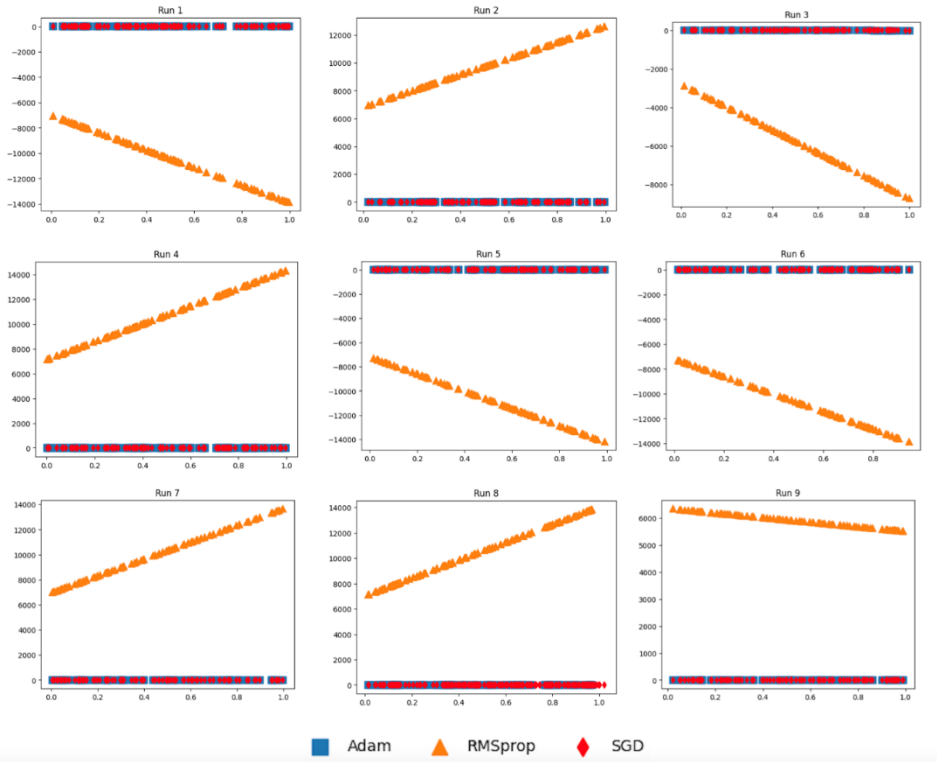

Figure 8: Plots for the three models as $(x_i, H_{\text{SGD}}(x_i))$, $(x_i, H_{\text{Adam}}(x_i))$, and $(x_i, H_{\text{RMSprop}}(x_i))$ for i=1..m according to x_i generated using F_2.

These results demonstrate conformance of the weight parameters of models utilizing Adam and SGD optimizers. Figure 8 reports the 2D plots of three induced models as $(x_i, H_{SGD}(x_i))$, $(x_i, H_{Adam}(x_i))$, *and* $(x_i, H_{RMSprop}(x_i))$ for i=1..m using data generated according to Equation 11 (i.e., F_2) [36]. It can be clearly seen that the data distributional difference of results via SGD is closer to that of Adam when compared to results obtained with the help of RMSprop. In Figure 11(b), we quantify distributional differences using Equations 14 and 15. It can be shown that Adam is closer to SGD as shown from AdamSGD when compared to that of RMSprop to SGD (i.e., RMSpropSGD) over the 9 runs. The quantification of AdamSGD is attributed to d_{SA} while RMSpropSGD is attributed to d_{SR}. In addition, the distributional differences between AdamSGD and RMSpropSGD are statistically significant (P-value = 7.01×10^{-7} from t-test).

Figures 9 and 10 report 2D plots of $(x_i, H_{SGD}(x_i))$, $(x_i, H_{Adam}(x_i))$, *and* $(x_i, H_{RMSprop}(x_i))$ for *i*=1..*m* using generated data of Equations 12 (F_3) and 13 (F_4) [37], where models were induced with SGD, Adam, and RMSprop optimizers. It can be seen from the alignment of Adam with SGD that Adam has a closer data representation to SGD compared to RMSprop to SGD. When quantifying the data distributional differences in Figure 11 (c and d), it can be clearly shown that the distributional differences of SGD and Adam (referred to AdamSGD) are closer than SGD to RMSprop over the 9 runs.

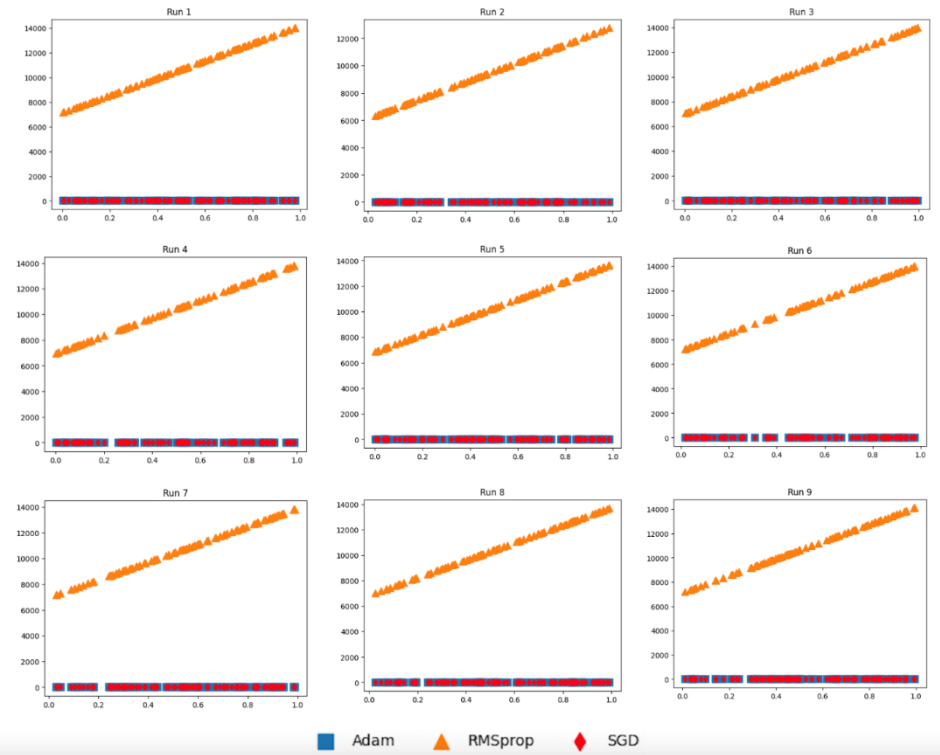

Figure 9: Plots for the three models as $(x_i, H_{\mathrm{SGD}}(x_i))$, $(x_i, H_{\mathrm{Adam}}(x_i))$, and $(x_i, H_{\mathrm{RMSprop}}(x_i))$ for *i*=1..*m* according to x_i generated using F_3.

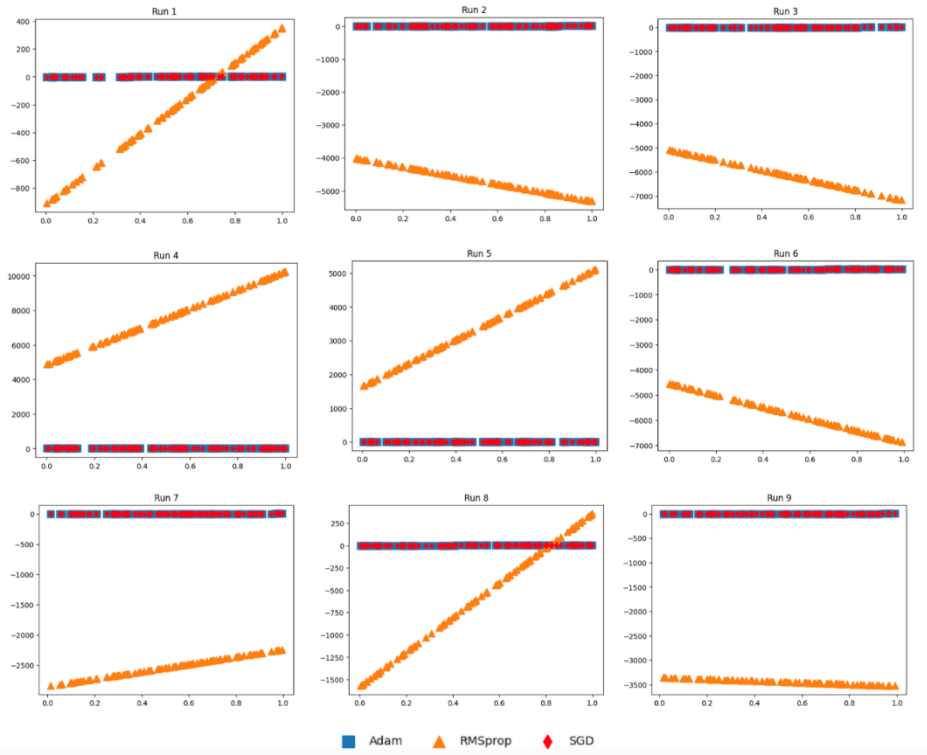

Figure 10: Plots for the three models as $(x_i, H_{\text{SGD}}(x_i))$, $(x_i, H_{\text{Adam}}(x_i))$, and $(x_i, H_{\text{RMSprop}}(x_i))$ for i=1..m according to x_i generated using F_4.

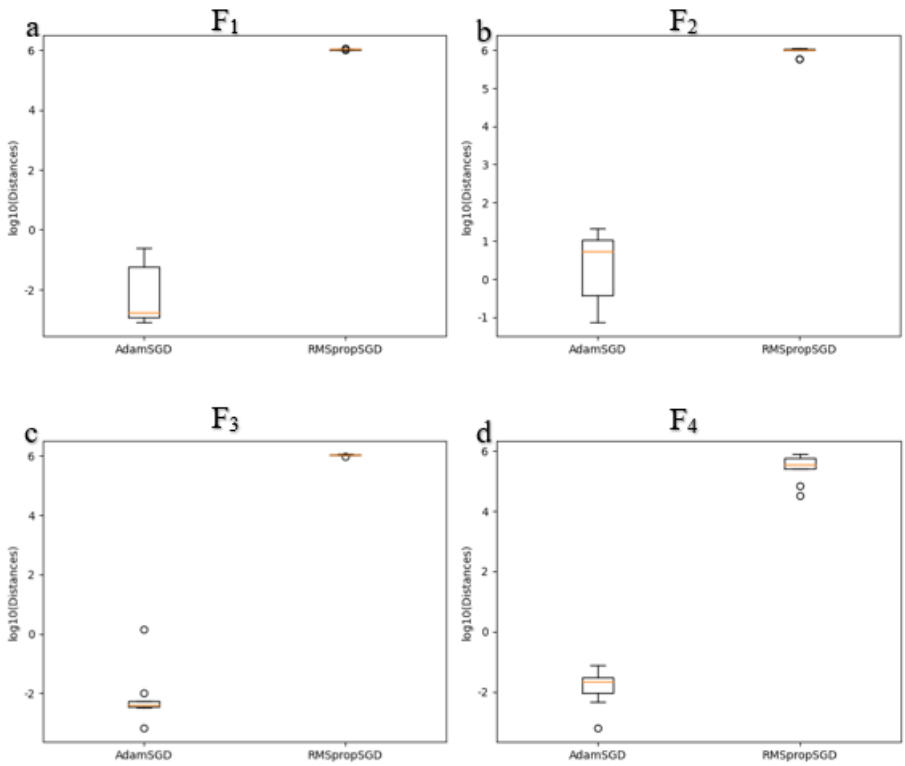

Figure 11: Boxplots of the four studied models, F_1-F_4, showing the distance distribution over nine runs for AdamSGD and RMSpropSGD. a) results for F_1. B) results for F_2. c) results for F_3. d) results for F_4.

These quantified results for AdamSGD and RMSprop are attributed to d_{SA} and d_{SR}, respectively. Additionally, the distributional differences of between AdamSGD and RMSprop were statistically significant (P-value = 3.48×10^{-12} from t-test when F3 is used while P-value = 1.49×10^{-3} from t-test when F4 is used). These results demonstrate the stable performance when SGD is coupled with Adam.

4. Discussion

Our deep transfer learning (DTL) models work as follows. In the TFe-based models, the convolutional base (also called the feature extraction part) in the pre-trained model is left unchanged while the densely connected classifier is modified to deal with the binary class classification at hand. Therefore, we applied the features extraction part of pre-trained models to the SCGRN images to extract features followed by a flattening step to train densely connected classifier from scratch. It can be noted that only weights of densely connected classifier are changed according to Adam optimizer while we transferred knowledge (i.e., weights) of the feature extraction part from pre-trained models. In terms of the TFt-based models, we keep weights of the bottom layers in the feature extraction part of pre-trained models unchanged while modifying weights in the proceeding layers including the densely connected classifier according to the Adam optimizer. Moreover, the densely connected classifier was altered to deal with the binary class classification problem pertaining to distinguishing between healthy controls and T2D SCGRN images. It can be seen that updating model weight parameters is done through the training with the use of Adam optimizer.

When conducting experimental study to assess deep transfer learning models, we used different optimizers, including SGD, RMSprop, and Adam. For SGD optimizer used in our DTL models, the model weight parameters were different than DTL models coupled with RMSprop and Adam optimizers. Therefore, we induced three sets of different models attributed to the three optimizers. Experimental results demonstrate the superiority of DTL models utilizing SGD and Adam optimizers when compared to that using SGD and RMSprop optimizers. We reported training loss for epochs pertaining to TFe-based and TFt-based models when running 5-fold cross validation in Supplementary Figure S1**.** For each DTL model, the number of layers including frozen and unfrozen layers is reported in Supplementary Table S1.

In our study, mitigation of overfitting is attributed to (1) transfer learning in which many layers in DTL models are freezed and thereby reducing the number of trainable parameters; and (2) applying the dropout to the last fully-connected layer in which we set the dropout rate to 0.5 [38]. It is worth mentioning that we assessed the performance of other deep learning (DL) models such as ConvNeXtTiny and ConvNeXtLarge [39]. Although ConvNeXtLarge outperformed ConvNeXtTiny, they didn't exhibit superior performance when compared to TFeSEResNeXT101. Therefore, we include their performance results in Supplementary Tables S2 and S3 (and Supplementary Figure S2). In terms of the running time, TFeSEResNeXT101was 1.54x faster than TFeConvNeXtLarge. We report running time for ConvNeXtTiny-based and ConvNeXtLarge-based models in Supplementary Figure S3.

In our DTL models, we have transferred weights from pre-trained models coupled with weights obtained with the help of Adam optimizer. If weight parameters obtained using two optimizers are close, then the two models almost behave the same. On the other hand, when the weight parameters resulted from two optimizers are not close, then the two models behave differently. To mimic the real scenario and investigate the effects of coupling different model weight parameters, we performed a simulated study. In Figures 7-11, we showed that a model induced with the help of SGD optimizer is closer to a model induced with Adam optimizer when compared to a model induced with the help of RMSprop optimizer. It can be evident from visualized results in our study that SGD and Adam had less distributional differences than that of SGD and RMSprop. That resembles the case of having two related datasets for SGD and Adam against unrelated datasets for SGD and RMSprop. As a result, inferior performance results for models utilizing RMSprop are attributed to the high distributional differences in model weight parameters.

It is worth noting that our DTL models keep weights of many layers unchanged. Therefore, when we trained our models, we had fewer number of updated weights compared to updated weights in models trained from scratch. It can be seen from Figure 6 that our DTL models are fast and can be adopted into mobile applications. It can be noticed from Tables 2 and 3 that leveraging source task knowledge contributed to improved prediction performance when coupled with updated weight parameters in the target task using Adam optimizer. On the other hand, the transferred knowledge from the source task contributed to degraded performance when coupled with updated weight parameters in the target task using RMSprop and SGD optimizers. Also, when we assessed

additional DL models such as ConvNeXtLarge and ConvNeXtTiny, the knowledge transfer contributed to maintain the same performance behavior in which leveraging source domain knowledge when coupled with updated weights in the target task remained to be the best (see Supplementary Tables S2 and S3).

5. Conclusions and Future Work

In this paper, we present and analyze deep transfer learning (DTL) models for the task of classifying 224 SCGRN images pertaining to healthy controls and T2D patients. First, we utilized seven pre-trained models (including SEResNet152 and SEResNeXt101) already trained on more than million images from the ImageNet dataset. Then, we left weights in the convolutional base (i.e., feature extraction part) unchanged and thereby transferring knowledge from pre-trained models while modifying the densely connected classifier with the use of Adam optimizer to discriminate heathy and T2D SCGRN images. Another presented DTL models work as follows. We kept weights of bottom layers in the feature extraction part of pre-trained model unchanged while modifying consequent layers including the densely connected classifier with the use of Adam optimizer. Experimental results on the whole 224 SCGRN image dataset using 5-fold cross-validation demonstrate the superiority of TFeSEResNeXT101, achieving the highest average BAC of 0.97 and therefore significantly surpassing the performance of the baseline resulted in an average BAC of 0.86. Furthermore, our simulation study showed that the highly accurate performance in our models is attributed to the distributional conformance of weights obtained with the use of Adam optimizer when coupled with weights of pre-trained models.

Future work includes (1) adopting our computational framework to analyze DTL models with different network topologies and thereby identifying the best practice for DTL; (2) incorporating multi-omics datasets with images to improve the prediction performance using DTL models; (3) developing a boosting mechanism to improve the performance of DTL models in different biological problems [40, 41]; (4) incorporating feature representation obtained via our DTL models with machine learning algorithms for the task of inferring SCGRNs; and (5) utilizing our framework to speed up the learning process, e.g., TFeVGG19 was 802.67x faster than VGG19, trained from scratch.

Author Contributions

T.T. conceived and designed the study. S.A. performed the deep learning experiments and the visualization of results. T.T. performed the analysis. T.T. and S.A. wrote the manuscript. T.T. supervised the study. All authors have read and agreed to the revised version of the manuscript.

Funding

This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, Saudi Arabia, under grant No. (D-087-611-1440). The authors, therefore, gratefully acknowledge the DSR for technical and financial support.

Data Availability

The dataset analyzed during the current study is available in the dataset folder within supplementary material at https://www.biorxiv.org/content/10.1101/2020.08.30.273839v1.supplementary-material. The single-cell gene expression data is available in the ArrayExpress repository under accession number E-MTAB-5061 (https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-5061).

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Hemerich, D., et al., *Effect of tissue-grouped regulatory variants associated to type 2 diabetes in related secondary outcomes.* Scientific Reports, 2023. **13**(1): p. 3579.
2. Xie, D., et al., *Global burden and influencing factors of chronic kidney disease due to type 2 diabetes in adults aged 20–59 years, 1990–2019.* Scientific Reports, 2023. **13**(1): p. 20234.
3. Parker, E.D., et al., *Economic costs of diabetes in the US in 2022.* Diabetes Care, 2024. **47**(1): p. 26-43.
4. Mohsen, F., et al., *A scoping review of artificial intelligence-based methods for diabetes risk prediction.* npj Digital Medicine, 2023. **6**(1): p. 197.

5. Su, X., et al., *Ten metabolites-based algorithm predicts the future development of type 2 diabetes in Chinese.* Journal of Advanced Research, 2023.
6. He, Y., et al., *Comparisons of polyexposure, polygenic, and clinical risk scores in risk prediction of type 2 diabetes.* Diabetes Care, 2021. **44**(4): p. 935-943.
7. Edlitz, Y. and E. Segal, *Prediction of type 2 diabetes mellitus onset using logistic regression-based scorecards.* Elife, 2022. **11**: p. e71862.
8. Kokkorakis, M., et al., *Effective questionnaire-based prediction models for type 2 diabetes across several ethnicities: a model development and validation study.* EClinicalMedicine, 2023. **64**.
9. Pyrros, A., et al., *Opportunistic detection of type 2 diabetes using deep learning from frontal chest radiographs.* Nature communications, 2023. **14**(1): p. 4039.
10. Kingma, D.P. and J. Ba, *Adam: A Method for Stochastic Optimization*, in *3rd International Conference on Learning Representations*. 2015: San Diego, CA, USA.
11. Wachinger, C., T.N. Wolf, and S. Pölsterl, *Deep learning for the prediction of type 2 diabetes mellitus from neck-to-knee Dixon MRI in the UK biobank.* Heliyon, 2023. **9**(11).
12. Das, B., *A deep learning model for identification of diabetes type 2 based on nucleotide signals.* Neural Computing and Applications, 2022. **34**(15): p. 12587-12599.
13. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
14. Naveed, I., et al., *Artificial intelligence with temporal features outperforms machine learning in predicting diabetes.* PLOS Digital Health, 2023. **2**(10): p. e0000354.
15. Bengio, Y., I. Goodfellow, and A. Courville, *Deep learning*. Vol. 1. 2017: MIT press Cambridge, MA, USA.
16. Wu, D., et al., *Multi-Feature Map Integrated Attention Model for Early Prediction of Type 2 Diabetes Using Irregular Health Examination Records.* IEEE Journal of Biomedical and Health Informatics, 2023.
17. Shu, H., et al., *Modeling gene regulatory networks using neural network architectures.* Nature Computational Science, 2021. **1**(7): p. 491-501.
18. Badia-i-Mompel, P., et al., *Gene regulatory network inference in the era of single-cell multi-omics.* Nature Reviews Genetics, 2023: p. 1-16.
19. Turki, T. and Y.h. Taguchi, *Discriminating the single-cell gene regulatory networks of human pancreatic islets: A novel deep learning application.* Computers in Biology and Medicine, 2021. **132**: p. 104257.
20. Iacono, G., et al., *bigSCale: an analytical framework for big-scale single-cell data.* Genome research, 2018. **28**(6): p. 878-890.
21. Iacono, G., R. Massoni-Badosa, and H. Heyn, *Single-cell transcriptomics unveils gene regulatory network plasticity.* Genome biology, 2019. **20**(1): p. 110.
22. Tripathi, S., M. Dehmer, and F. Emmert-Streib, *NetBioV: an R package for visualizing large network data in biology and medicine.* Bioinformatics, 2014. **30**(19): p. 2834-2836.
23. Simonyan, K. and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, in *3rd International Conference on Learning Representations (ICLR)*. 2015: San Diego, CA, USA.
24. Chollet, F. *Xception: Deep learning with depthwise separable convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
25. Huang, G., et al. *Densely connected convolutional networks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

26. Segerstolpe, Å., et al., *Single-cell transcriptome profiling of human pancreatic islets in health and type 2 diabetes.* Cell metabolism, 2016. **24**(4): p. 593-607.
27. Szegedy, C., et al. *Rethinking the inception architecture for computer vision*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
28. He, K., et al. *Identity mappings in deep residual networks*. in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. 2016. Springer.
29. Hu, J., L. Shen, and G. Sun. *Squeeze-and-excitation networks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
30. Bottou, L., *Stochastic gradient descent tricks*, in *Neural Networks: Tricks of the Trade: Second Edition*. 2012, Springer. p. 421-436.
31. Chollet, F., *Deep Learning with Python*. 2017, Manning Publications Co.
32. Team, R.C., *R: A Language and Environment for Statistical Computing.* Journal of Statistical Software, 2008. **25**(1).
33. Franco, V.R., *optimg: General-Purpose Gradient-Based Optimization*. 2021.
34. Hunter, J.D., *Matplotlib: A 2D graphics environment.* Computing in science & engineering, 2007. **9**(03): p. 90-95.
35. Ruder, S., *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.
36. Currin, C., et al., *A Bayesian approach to the design and analysis of computer experiments*. 1988, Oak Ridge National Lab., TN (USA).
37. Forrester, A., A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. 2008: John Wiley & Sons.
38. Gao, H., J. Pei, and H. Huang. *Demystifying dropout*. in *International Conference on Machine Learning*. 2019. PMLR.
39. Liu, Z., et al. *A convnet for the 2020s*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022.
40. Turki, T. and Z. Wei, *Boosting support vector machines for cancer discrimination tasks.* Computers in biology and medicine, 2018. **101**: p. 236-249.
41. Turki, T. and Z. Wei, *Improved Deep Convolutional Neural Networks via Boosting for Predicting the Quality of In Vitro Bovine Embryos.* Electronics, 2022. **11**(9): p. 1363.