kNN-SVM with Deep Features for COVID-19 Pneumonia Detection from Chest-Xray

Aman Bahuguna[1], Deepak Yadav[1], Apurbalal Senapati[2], and Baidya Nath Saha[3]*

[1] Chandigarh University, Punjab, 140413, India {amanbh4637, deepak842705}@gmail.com
[2] Central Institute of Technology, Kokrajhar, Assam, 783370, India a.senapati@cit.ac.in

[3] Concordia University of Edmonton, Alberta, T5B 4E4, Canada baidya.saha@concordia.ab.ca

Abstract. Most attention has been paid to chest Computed Tomography (CT) to encounter COVID-19 pandemic because high-resolution CT can demonstrate visual signatures of respiratory illness that causes lung damage for many COVID-19 patients. However, CT is very expensive, time consuming, and inaccessible in remote hospitals. As an important complement, this research proposes a more economically and practically feasible solution by offering a novel kNN regularized Support Vector Machine (kNN-SVM) algorithm for identifying COVID-induced pneumonia from frontal chest X-ray (CXR). To compute the deep features, we used transfer learning on the standard VGG16 model. Then the autoencoder algorithm is used for dimensionality reduction. Finally, a novel kNN-regularized Support Vector Machine algorithm is developed and implemented which can successfully classify the three classes: Pneumonia, COVID-19, and normal on a benchmark dataset of chest x-ray. kNN-SVM combines the properties of two well-known formalisms: k-Nearest Neighbors (kNN) and Support Vector Machines (SVMs). Our approach extends the total-margin SVM, which considers the distance of all points from the margin; each point is weighted based on its k nearest neighbors. The intuition is that examples that are mostly surrounded by similar neighbors, i.e., of their own class, are given more priority to minimize the influence of drastic outliers and improve generalization and robustness. Thus, our approach combines the local sensitivity of kNN with the global stability of the total-margin SVM. Extensive experimental results demonstrate that the proposed kNN-SVM can detect COVID-19 induced pneumonia from chest x-ray with greater or comparable accuracy relative to human radiologists.

Keywords: kNN SVM COVID-19 Pneumonia Chest X-ray Deep Learning VGG16

* Corresponding author.

1 Introduction

The recent outbreak of novel coronavirus disease, COVID19, has had an unprecedented impact on global healthcare systems [28]. Many COVID-19 infections have included respiratory illness manifesting as fever and cough, developing pulmonary symptoms such as chest discomfort and shortness of breath, and clinically resembling viral pneumonia with the hallmark features of COVID-19 infection as bilateral, peripheral ground-glass, and consolidative pulmonary opacities on CT [9]. High resolution CT, which combines multiple X-ray images from different angles into a single image, can image detailed visual signatures. Unfortunately, a CT scan to manage the pandemic is not practical because it is expensive, time-consuming, labor-intensive, inaccessible in remote hospitals, scanning equipment needs prolonged deep sterilization after each potentially infected patient is scanned and there is always the risk of transmission of the virus to healthcare workers [4]. To better address these issues, CXR, though less sensitive to detect the lung pathology caused by the coronavirus has taken center stage as a front line diagnostic test because X-ray machines are commonly available, scans are inexpensive, and they are found in both emergency and rural hospitals, can be installed on a mobile platform, relatively easy to disinfect, and are one of the most affordable ways to respond the outbreak [32].

Deep learning has been widely used in medical imaging over the last decade, and it has often outperformed the performance of medical professionals [35]. Finding the presence of pneumonia in the chest X-ray can be interpreted as a classification problem. Several Convolutional Neural Networks (CNNs) based deep learning models show great performance on various image classification tasks, VGG16 is one of them. However, deep learning models rely on large-scale datasets to train and evaluate classifiers. Due to the scarcity of COVID-19 CXR samples, transfer learning is preferred in this context. In this study, we used the VGG16 model from Keras, which was pretrained on a large scale ImageNet dataset. Transfer learning avoids the need to train deep models from scratch, as well as the lack of training data, it takes advantage of the extraction of knowledge achieved through visual recognition from large scale ImageNet.

To improve the performance of VGG16 model, this study performs four sequential steps: (a) first we extract the deep or bottleneck features from the VGG16 model's second last dense layer; (b) then reduce the dimension through Autoencoder algorithm; and (c) cluster the reduced features through K-Means algorithm and aggregate the cluster information in the reduced feature sets; and finally (d) classify through kNN-SVM. Experimental results demonstrate that clustering information improves the performance of classification results.

SVMs [19] have been widely used in many applications due to their robustness, especially for small training sets, adaptability to various classification and regression problems through the incorporation of appropriate kernel functions, as well as the ability to obtain a global optimal solution via quadratic programming. However, it is well-known that minimizing solely the empirical training error can result in poor generalization due to overfitting [47]. Regularization methods such as 1-norm [51] or manifold regularization [7], and loss

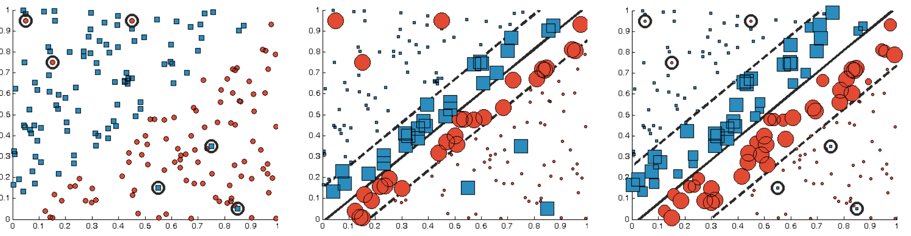


Fig. 1: A simple synthetic example to compare the behavior of the proposed *k*NN-SVM (right) with classical soft-margin SVM (center). The data set (left) is almost linearly separable except for the six circled outliers. These outliers receive high weights under the classical soft-margin SVM. In contrast, *k*NN-SVM minimizes their effect as they have few neighbors of the same class as themselves. As a result, the effects of overfitting are reduced, and the *k*NN-SVM can **achieve a greater margin**. We show theoretically and empirically that this corresponds to better generalization.

functions such as least-square SVM [41] have been proposed as a solution to this problem. However, a persistent problem endemic to these approaches is their inability to perform under noisy conditions. This issue is especially exacerbated when these methods are confronted with extreme outliers. As the approaches are margin-based, they tend to weigh the outliers very strongly, which leads to overfitting and a loss of robustness; this ultimately affects generalization.

To mitigate the effects of outliers and inspired by previous attempts to weigh individual examples differently during training, we propose to use a weighting scheme based on *k*NN [14, 20]. We propose to assign weights to training examples proportional to the number of neighbors of the same class. The intuition is that, locally, a data point will be surrounded by similar neighbors (i.e., of the same class), and consequently, extreme outliers will be weighted less. This prevents such outliers from exerting too much influence on the final classifier and improves robustness. However, depending on the choice of k and the density of the data, *k*NN itself can be very locally sensitive. This motivates the adoption of total-margin SVMs [49] as the formalism underlying our approach. The total-margin SVM extends classical SVMs by adding extra surplus terms in the objective and constraints that measure the data point deviation from the classification hyperplane [31, 26]. Thus, while the slack variables measure the deviation of miscategorized points, the surplus variables measure the deviation of the correctly categorized points. Training a classifier that maximizes the *total margin* requires minimizing error (measured by slack variables) and maximizing "right classification" (measured by surplus variables). By weighting the data points in the total-margin SVM with *k*NN-based weights, our method combines the kNN's local sensitivity with the total-margin SVM's global stability. We refer to this algorithm as *k*NN-weighted SVM. Finally, *k*NN-SVM has the benefit of allowing the use of a wide range of different of distance metrics including those learned via metric learning approaches such as LMNN [46] and MDML [24].

On a synthetic example, Figure 1 compares the behavior of our proposed *k*NN-weighted SVM to that of classical SVMs. The data set (Figure 1, left) consists of uniformly-randomly generated linearly-separable data. The data set also contains six outliers, three for each class, which are circled. Figure 1 (center) shows the weights of the training examples for a standard soft-margin SVM. More importantly, for soft-margin SVM, the badly misclassified outliers have the same weights relative to the correctly classified within-margin examples. Figure 1 (right) shows the weights of the training examples for the kNN-weighted SVM. Because kNN-weighted SVM assigns weights proportional to similar neighbors, the outliers' influence on the classifier is greatly reduced. The reduced overfitting enables the maximization of the total margin. As shown in Figure Figure 1, the kNN-weighted SVM has a larger margin than the classical SVM.

There has previously been research on combining the power of kNNs and SVMs [18, 11, 37, 25]. These strategies all seek to "localize" by picking a few training examples that are close to a test example and designing a SVM for these chosen training examples [18]. At a high level, these methods can be thought of as learning one SVM for each partition of the data space. The class of these approaches in general, and the SVM-KNN approach [50] in particular, is closely related to our approach where it finds the neighbors to a query and then trains a local SVM. On the contrary, we train the SVM on all the examples from the training set by incorporating the kNN distance function directly into our optimization problem. Our method "localizes" SVMs by employing a data-driven regularization approach. Instead of focusing on a test point, we weigh the input training space based on their feature locations. Instead of using multiple localized SVM models [50], we use a single SVM model to capture both "global" and "local" information.

2 Previous Efforts on Deep learning for Covid pneumonia detection from chest X-ray

Islam *et al.* [22] reviewed different Artificial Intelligence (AI) especially machine learning algorithms and demonstrates how they combat against COVID-19 pandemic in the process of detection, screening, diagnosis, the progression of severity, mortality, drug repurposing, and other tasks by analyzing heterogeneous data consisting of digital imaging, clinical and laboratory results. Panwar *et al.* [33] created a deep model - "nConvNet" which employs transfer learning on pre-trained VGG16 for fast COVID-19 detection from CXR images. Similarly, Das *et al.* [16] utilized CNNs and Xception model to build deep transfer learning based COVID-19 detection model from chest X-rays. Bassi *et al.* [6] proposed a novel twice transfer learning method called "output neuron keeping" which performs better than both twice and simple transfer learning and simple transfer learning model. Twice transfer learning method is a two-stage simple transfer learning procedure where it trains the pretrained VGG16 model on a large CXR dataset first and then trains it with a smaller COVID-19 CXR dataset. Unlike twice transfer learning, this method keeps the output neurons (normal and pneumonia) of the first step (training on large chest x-ray) in the last step while training on COVID-19. This method adds only one output neuron for covid in the last layer pre-built with two output neurons: normal and pneumonia in the previous stage. They used Layer-wise Relevance Propagation (LRP) for improving the explainability of the deep neural network used. Brunese *et al.* [10] developed a two-stage transfer learning approach by pre-trained VGG16 model. In the first stage, they predict whether a CXR is of a healthy patient or one with generic pulmonary disease. Then, if this image is of a generic pulmonary disease patient, they pass this image to another model which detects whether this pulmonary disease is COVID-19 or not. Salman *et al.* [36] exploited the pretrained InceptionV3 model for feature extraction using transfer learning for COVID-19 Pneumonia detection. Jaiswal *et al.* [23] created a new model called "CovidPen" that detects COVID-19 infection from CXR images using transfer learning on the Pruned EfficientNet model. Pham [34] *et al.* reported that three pretrained CNN Models named AlexNet, GoogLeNet and Squeezenet demonstrate higher accuracy for COVID-19 classification from CXR taking lesser time as compared to pre-trained models. Al-Waisy et al. [2] devised a new deep learning system called Covid-CheXNet by combining two different deep learning models, ResNet34 and HRNet and utilizing the CLAHE method and the Butterworth bandpass filter to improve the poor image quality and reduce the noise level respectively. Waheed *et al.* [44] proposed a special type of network called CovidGAN which uses Auxiliary Classifier Generative Adversarial Network (AC-GAN) to generate synthetic CXR images. This research showed that the training dataset augmented with CovidGAN offers better classification results with deep neural networks. Ahmed *et al.* [1] proposed ReCoNet, an end-to-end CNN architecture that used two loss functions – Multi-tasking learn loss function and a joint weighted cross-entropy loss function for improving its performance. LV *et al.* [27] proposed cascade-SEMEnet which employed SEME-ResNet50 to detect the type of lung infection and a DenseNet169 to subdivide viral pneumonia. This study also used Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve contrast and U-Net to remove the non-pathological features from the CXR images.

3 Proposed Methodology for COVID Induced Pneumonia Detection from Chest X-ray

To improve the performance of transfer learning of pre-trained VGG16 model, this study explores a new avenue which follows the four sequential steps which are demonstrated in four different subsections:

3.1 Deep Feature Extraction Using VGG16

We used VGG16 [39] for extracting deep features from the CXR images. VGG16 is a 16-layer convolutional neural network. We used a pre-trained version of VGG16 trained on the ImageNet dataset [17]. Figure 2 illustrates the architecture and transfer learning on pre-trained VGG16 model for COVID-19 Pneumonia detection which has been used in this study to compute deep or bottleneck features. VGG16's architecture is very simple, consisting of convolution layers of

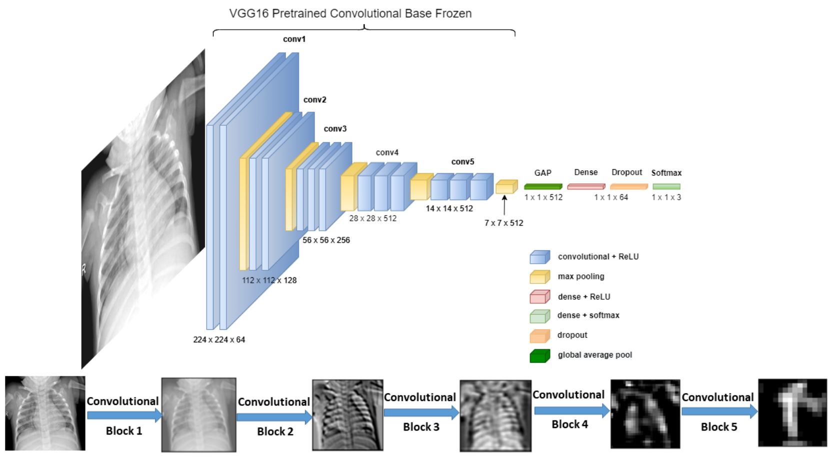


Fig. 2: VGG16 Model for COVID-19 Pneumonia Detection.

3x3 filter with stride 1 that always use the same padding and max pool layer of 2x2 filter with stride 2. This convolution and max pool arrangement is repeated throughout the architecture. Finally, for multiclass classification, it has two fully connected layers followed by a softmax [39]. Using pretrained VGG16 as a fixed feature extractor, we utilize the transfer learning approach. We load VGG16 network with weights pretrained on ImageNet, only keeping the convolutional base and truncating the fully connected layer head. Then, on top of the VGG16 convolutional base, we build our new fully-connected layer head, which includes a Global Average Pooling layer, a Dense layer with 64 neurons and ReLU activation, a Dropout layer with a 0.2 rate, and an output layer with softmax activation and 3 neurons for classification. The convolutional base of VGG16 is then frozen, so that only the fully connected layer head is trained. After the model has been trained, we pass the entire training and test data through the model and collect the deep features from the last Dense layer before the output layer.

3.2 Dimensionality Reduction Using Autoencoders

To reduce the dimensionality of the extracted deep features, we exploited Autoencoder [45], a self-supervised deep learning model. Autoencoder has two parts. Firstly, an encoder which is a compression unit that compresses the input data. And secondly, a decoder which decompresses the compressed input by reconstructing it. Fig. 3 depicts the architecture of the autoencoder used in this experiment. Except for the last, each Dense layer is followed by a BatchNormalization layer and a LeakyReLU activation layer with alpha = 0.3. The last Dense layer uses linear activation function. We trained the autoencoder with Adam as an optimizer and the learning rate of 0.001, Mean-Squared Error (MSE) as loss function, and batch size of 16. The autoencoder model is trained for 20 epochs

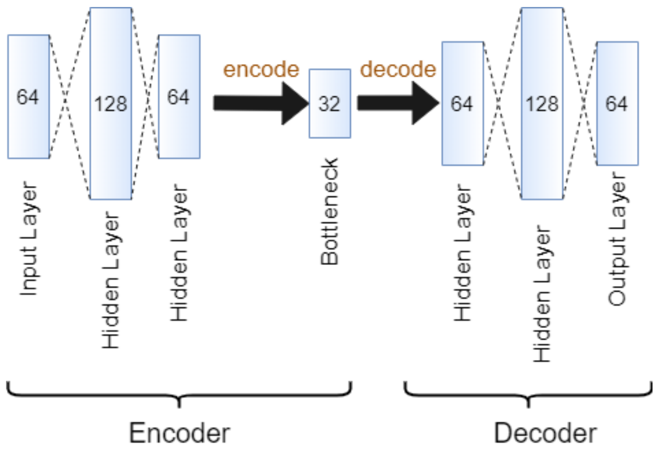
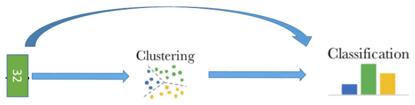


Fig. 3: Autoencoder Algorithm for dimensionality reduction

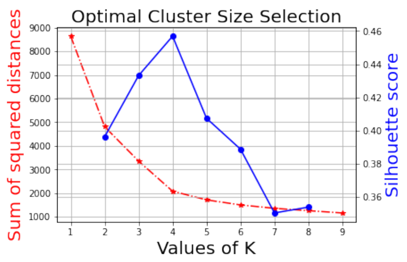
and post-training, we save its encoder part. We fed the deep features of train and test data into the encoder to reduce the feature space from 64 dimensions to 32 dimensions.

3.3 Incorporating clustering information into classification tasks

To improve the performance of the classification tasks, we cluster the reduced features computed above using the K-means algorithm [19] as illustrated in Fig. 4. K-means clustering divides n observations into k clusters, with each observation belonging to the cluster with the closest mean (cluster centers or cluster centroid). Optimal value of K is determined in this study using the Silhouette Method in combination with the Elbow Method.



(a) Clustering before Classification.



(b) Optimal Values of K for K-Means Algorithm.

Fig. 4: Incorporating clustering for improving classification tasks.

Elbow Method In the elbow method, we run K-means clustering on the reduced deep features for values of K ranging from 1 to 9. We compute average distances to the centroid across all data points for each of these K values. Then we plot these points and look for the point where the average distance from the centroid suddenly decreases. This point is called the Elbow which is the number of ideal clusters according to the Elbow method. The plot in Fig. 4b shows that $K = 4$ is the ideal value provided by the Elbow method.

Silhouette Method In the silhouette analysis, we use the silhouette coefficient [8] which compares how similar a data point is within cluster (cohesion) to other clusters (separation). The Silhouette coefficient $S(i)$ of an i'th point is given by, $S(i) = b(i)–a(i)/\max\{b(i), a(i)\}$, where

– $a(i)$ represents the average distance between i and other points within the same cluster.
– $b(i)$ represents the average distance between i and other points within the cluster closest to its cluster.

We calculate the silhouette coefficient for values of K from 2 to 8 as shown in Fig. 4b. We average the silhouette coefficients at each point to produce the average silhouette score, which is denoted as average Silhouette Score = Mean$S(i)$. At $K = 4$, we find that the silhouette score is maximum after plotting the average silhouette score against each value of K, as shown in Fig. 4b.

We finalize optimal number of cluster K as 4 using the information from the previous two methods. The reduced deep features along with cluster information we fed into a novel kNN-SVM demonstrated in the next subsection.

3.4 *k*NN regularized Support Vector Machine (*k*NN-SVM)

Let the training data consists of l pairs $(\boldsymbol{x}_i, y_i), i = 1, ..., l$, with $x_i \in \mathrm{I\!R}^p$ and $y_i \in \{-1, 1\}$. Typically in SVMs, the mapping function $\phi(\boldsymbol{x})$ or kernel maps the input space into a high dimensional feature space aiming to enhance the linear separability. The following Quadratic Programming (QP) problem [19] is used to find the optimal separating hyperplane of the kNN-SVM:

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{w}||_2^2 + C\sum_{i=1}^{l} D_i\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^t\boldsymbol{z}_i + b) \geq D_i(1 - \xi_i), \forall i, \xi_i \geq 0. \end{aligned} \tag{1}$$

where, $\boldsymbol{z}_i = \phi(\boldsymbol{x}_i)$, $\boldsymbol{w}$ and C refer to a weight vector and the margin parameter that determines the trade-off between the margin maximization$(2/||\boldsymbol{w}||)$ and error minimization respectively. $\xi_i(i = 1, ..., l)$ are non-negative variables called slacks which measure the distance of the example $(\boldsymbol{x}_i, y_i)$ from the optimal separating hyperplane. $D_i = \frac{|N_i|}{k}$, and $N_i \in \{N(i) : C(N(i)) = C(i)\}$, where $C(i)$ and $N(i)$ are the class and the neighbors of i respectively. Given that we are using a *k*NN formulation, $N(i) = k,\ \forall i$. D_i explains the significance of the slack and surplus variables. The key intuition in this framework is that instances surrounded by instances in the training data from its own class get more importance than the instances surrounded by the members of the opposite class. For traditional SVM, $D_i = 1,\ \forall i$. We now present some key features of this framework as a function of the number of neighbors (k).

– It is necessary that $k \geq 1$.
– When $k == l$, where l is the number of training instances, every D_i becomes the fraction of the number of examples in its class. Then this formulation reduces to the total margin SVM as proposed by Yoon et al. [49].
– If k is sufficiently large, the formulation is robust to outliers and noise and works well.
– If k is quite small, say $k \longrightarrow 1$, the formulation can become potentially infeasible. This is due to the fact that outliers can have their D values to be 0 (as no neighbors of their class may be within k). To avoid such cases, we perform a pre-processing step and identify the number of neighbors that will mitigate the effects of the outliers.
– While it is potentially possible to add a small non-zero term, say ϵ, to each D_i, it does not work well. First when $\epsilon \longrightarrow 0$, the problem can become infeasible. But if ϵ is higher, the final solution can become very sensitive to the choice of this parameter. A more principled way of defining D_i is to employ Laplace correction, a standard method for probability estimation to ensure that the probabilities do not go to 0 or 1. Hence, $D_i = \frac{|N_i|+1}{k+n}$ where n is the number of classes. This will ensure that outliers get a weight of 1/2 while the rest of the points will end up with a reasonable weight.

We show empirically in a few domains that as long as the value of k is reasonable, our method is quite robust. Typically, we introduce Lagrange multipliers, α_i, and obtain the dual:

$$\begin{aligned} &\max_{\alpha_i} \sum_{i=1}^{l} \alpha_i D_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\text{subject to} \quad 0 \leq \alpha_i \leq C, \forall i, \quad \sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \tag{2}$$

where, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j)$ is any kernel function. Solving the above dual problem, we obtain the decision function needed to predict the classification of a new data point $\boldsymbol{x}'$: $f(\boldsymbol{x}') = sign(\sum_{i=1}^{l} \alpha_i^* y_i K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*)$, where, '*' denotes the optimal solution. The value of b^* can be obtained from the constraint of Equation 1 using Karush-Kuhn-Tucker (KKT) conditions: $b^* = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m K(\boldsymbol{x}_m, \boldsymbol{x}_s))$, where S and N_s represent the set of indices of support vectors and the number of support vectors respectively.

Theoretical Analysis - Margin Distribution Bounds on Generalization To obtain margin distribution bounds for *k*NN-SVM we can simply adapt the earlier proved theorem [49] for total margin SVMs. In order to make our derivation consistent with published literature, we adapt their notations and follow their theorem to derive ours. We first present the theorem and then discuss the similarities and differences to the original bounds next.

Theorem 1 *let $b \in \Re$ be fixed. Now an unknown but fixed probability distribution can be assumed in the input space $\chi \times \{-1, 1\}$ and the support in the ball of radius R can be considered about the origin χ. Then training sets $S = (X, Y)$ of size l can be drawn with probability $1 - \delta$ for all $\gamma > 0$ such that $\zeta((\boldsymbol{x}, y), \boldsymbol{w}, \gamma) = \eta((\boldsymbol{x}, y), \boldsymbol{w}, \gamma) = 0$, for some $(\boldsymbol{x}, y) \in S$, the bound of the generalization error of kNN weighted SVM is defined by*

$$\epsilon(l, \kappa, \delta) = \frac{2}{l}(\kappa \log_2(\frac{8el}{\kappa}) \log_2(32l) + \mathcal{O}(\log_2(\frac{l \log_2 l}{\delta}))),$$

where $\kappa = \lfloor \frac{65[(R+\boldsymbol{\zeta}/\boldsymbol{\eta})^2 + 0.5R\boldsymbol{\zeta}/\boldsymbol{\eta}]}{\gamma^2} \rfloor$, for $\boldsymbol{\zeta} = \boldsymbol{\zeta}(S, \boldsymbol{w}, \gamma), \boldsymbol{\eta} = \boldsymbol{\eta}(S, \boldsymbol{w}, \gamma)$ and provided $l \geq 2/\epsilon$, $\kappa \leq el$.

It is straightforward to prove theorem 1. The similar theorem for SVM and its proof is available in [49]. The bounds presented above have some attractive properties. The error increases monotonically with the value of κ with fixed l (number of examples) and δ, as in the total margin case. Note that as with the earlier case, κ depends on the number of slack variables. It increases with the slack variables. Hence, the bounds are affected primarily by the points that are extreme outliers (with a very small number of neighbors of the same class). In general case, this is a fairly robust bound as with the total margin case.

Multiclass *k*NN-weighted SVM We now extend the QP formulation of *k*NN weighted SVM for two-class classification into general multiclass settings [15]. Assume the given training data consists of l pairs $(\boldsymbol{x}_i, y_i), i = 1, ..., l$ with $\boldsymbol{x}_i \in \mathbb{R}^p$, where a label y_i is assigned for each example from a fixed finite set $\mathcal{Y} \in \{1, ..., m\}$. A feature function $\psi(\boldsymbol{x}, \boldsymbol{y})$ can be defined in such a way that it explicitly includes the y-labels, and for each class it allows a distinct weight vector $\boldsymbol{w}_k$ k [42]. We define $\psi(\boldsymbol{x}, y)$ for our experiment as,

$$\psi(\boldsymbol{x}, y) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_y)' \boldsymbol{\Sigma}_y^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_y)) \tag{3}$$

We now consider the following Quadratic Programming (QP) based optimization problem [43] to define the multiclass *k*NN-SVM.

$$\begin{aligned} &\min_{w, \boldsymbol{\zeta}, \boldsymbol{\eta}} \frac{1}{2} ||\boldsymbol{w}||_2^2 + C \sum_{i=1}^{l} D_i \zeta_i \\ \text{subject to} \quad &\boldsymbol{w}^t \delta\psi_i(\boldsymbol{y}) \geq D_i(1 - \zeta_i), \\ &\forall i, \forall y \in \mathcal{Y} \setminus y_i, \zeta_i \geq 0, \eta_i \geq 0. \end{aligned} \tag{4}$$

Algorithm 1 *k*NN-SVM Algorithms for both binary and multiclass classification

```
**kNN-SVM Training**
**Input:** $l$ labeled example $\{(x_i, y_i)\}_{i=1}^{l}$
**Output:** Estimated function $f : \Re^n \rightarrow \Re$
**Procedure:**
 **Step1:** Construct $D_i, i = 1, .., l$ using $D_i = |N_i|/|N(i)|$, and $N_i \in \{N(i) : C(N(i)) = C(i)\}$.
 **Step2:** Choose $\phi(\boldsymbol{x})$ for binary and $\psi(\boldsymbol{x}, y)$ for multiclass. Then compute kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$
 **Step3:** Choose $C$ and compute $\boldsymbol{\alpha}$. If binary, compute $b^*$ else compute $\boldsymbol{w}$

**kNN-SVM Test**
**Input:** A new test data point $\boldsymbol{x}'$
**Output:** Prediction class of $\boldsymbol{x}', f(\boldsymbol{x}')$
Return $f(\boldsymbol{x}') = sign(\sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*)$(**binary classification**) or
$f(\boldsymbol{x}') = \arg\max_y \boldsymbol{w}^t \psi(\boldsymbol{x}', \boldsymbol{y})$ (**multiclass classification**)
```

$\delta\psi_i(\boldsymbol{y})$ is defined as, $\delta\psi_i(\boldsymbol{y}) \equiv \psi(\boldsymbol{x}_i, y_i) - \psi(\boldsymbol{x}_i, \boldsymbol{y})$. Variables ζ_i, and D_i are defined above. Once a complete weight vector is learned, a new test example $\boldsymbol{x}'$ is classified as, $f(\boldsymbol{x}') = \arg\max_y \boldsymbol{w}^t \psi(\boldsymbol{x}', y)$ [48]. The subsequent dual formulation is:

$$\begin{gathered}\max_{\boldsymbol{\alpha}} \sum_{i,\boldsymbol{y}\neq y_i} \alpha_{i\boldsymbol{y}} D_i - \frac{1}{2} \sum_{\substack{i,\boldsymbol{y}\neq y_i \\ j,\bar{\boldsymbol{y}}\neq y_j}} \alpha_{i\boldsymbol{y}} \alpha_{j\bar{\boldsymbol{y}}} \delta\psi_i^t(\boldsymbol{y}) \delta\psi_j(\bar{\boldsymbol{y}}) \\ \text{subject to } \alpha_{i\boldsymbol{y}} \leq C, \forall i, \forall y \in \mathcal{Y} \setminus y_i\end{gathered} \tag{5}$$

Algorithm for *k*NN-SVM Algorithm 1 provides the pseudocode for the proposed *k*NN-SVM. As can be seen, in the first step, we calculate the distances based on the *k*NN. i.e., we construct the vector, $D_i, i = 1, .., l$ using k nearest neighbors where $D_i = |N_i|/|N(i)|$, and $N_i \in \{N(i) : C(N(i)) = C(i)\}$, where $C(i)$ and $N(i)$ represent the class and the neighbors of i respectively. For this step, suitable distance metric such as, LMNN [46], LFDA [40], DistBoost [21] or euclidean distance can be selected through cross-validation. In the next step, we choose the mapping function according to whether the classification is binary or multiclass i.e., we choose the mapping function $\phi(\boldsymbol{x})$ and then compute the kernel, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j)$ (**binary classification**) or choose $\psi(\boldsymbol{x}, y)$ defined in Equation 4 (**multiclass classification**). The appropriate kernel can then be computed. In the final step, we choose C. Then we compute $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_l)$ defined above and then compute $b^* = \frac{1}{|S|} \sum_{s\in S} (y_s - \sum_{m\in S} \alpha_m y_m K(\boldsymbol{x}_m, \boldsymbol{x}_s))$, where S denotes the set of the support vectors (**binary classification**). In the case of multiclass classification, we compute $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_l)$ using Equation 5 and then compute $\boldsymbol{w} = \sum_{i,\boldsymbol{y}\neq y_i} \alpha_{i\boldsymbol{y}}[\psi(\boldsymbol{x}_i, y_i) - \psi(\boldsymbol{x}_i, \boldsymbol{y})]$ for each class k. The function kNN-SVM Test outlines the code for the classification of a test instance.

4 Experimental Results and Discussions

4.1 Dataset Description

The dataset used consists of CXR images of COVID-19 patients, pneumonia patients and normal people. It includes 69 confirmed COVID-19, 79 confirmed pneumonia, and 158 normal images, i.e., overall 306 images. Again, the pneumonia images consist of 79 bacterial pneumonia and 79 viral pneumonia cases. Images have been accumulated in this dataset from diverse sources. The normal and pneumonia images are collected from the Kaggle CXR dataset [30]. This dataset consists of CXR images (anterior-posterior) selected from retrospective cohorts of pediatric patients aged one to five years old at Guangzhou Women and Children's Medical Center, Guangzhou. In addition to Kaggle, the COVID-19 X-ray images are collected from the COVID-19 CXR dataset [13], organized by Dr Joseph Paul Cohen. This is a publicly available dataset of CXR images from patients who have COVID-19 or other viral or bacterial pneumonia. Both of these datasets contain posterior-anterior chest images of pneumonia patients. The data is split into training and test category which is demonstrated in the Table 1.

Table 1: Training and test data distribution

Dataset	COVID-19	Normal	Pneumonia	
			Bacterial	Viral
Train	60	70	70	70
Test	9	9	9	9

4.2 Exploratory Data Analysis (EDA)

Fig. 5 shows the mean, standard deviation, mean of the remaining classes, difference between the mean of a particular class and the remaining classes (all other classes except the particular class), and Eigen image of the normal, pneumonia and COVID-19 patient respectively. COVID-19 images are distinguished from other classes in the middle and lower zone, as shown in Fig. 5, whereas pneumonia images are discriminated from other classes in the upper zone, and for normal classes, lung boundaries are predominant without structural deformities.

4.3 Image Pre-processing, Data Augmentation, and Transfer Learning with VGG16 Model

The CXR images are pre-processed before feeding to VGG16. Because VGG16 typically accepts input images of size 224X224, Chest X-ray images are resized

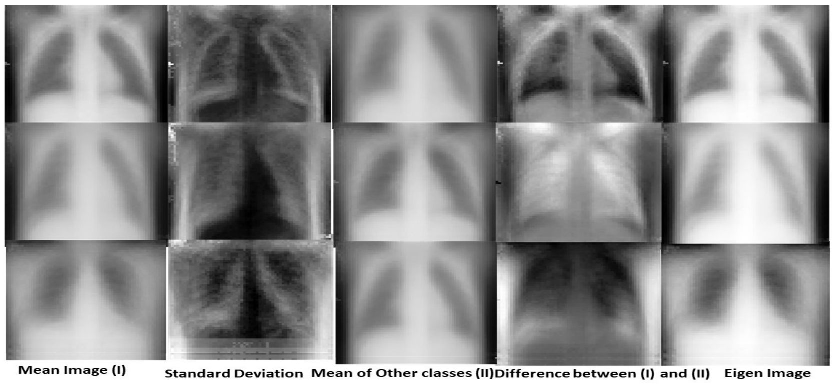


Fig. 5: Exploratory Data Analysis on chest x-ray images for three classes. Top, middle, and bottom row demonstrate for Normal, Pneumonia, and COVID-19 classes respectively.

to 224X224 while keeping the RGB channels intact in this study. Deep Neural Networks need a large amount of data for effective training. Otherwise, they may easily overfit on training data and give less accuracy on test data. But it is not possible to have a large dataset size in every medical imaging problem. Also, collecting such a medical image dataset may sometimes be time consuming and expensive. To enhance the model's performance and aid in model generalization, we perform data augmentation on training data using Keras ImageDataGenerator module [12].

For training VGG16, Adam optimizer with learning rate of 0.001, categorical cross-entropy loss function, and batch size of 16 is used. In this experiment, the model is trained end-to-end for 50 epochs. This model achieves a validation accuracy of 90.24% and a test accuracy of 75%. The training is executed using the Keras library [12]. The collected features of the training and test image data are then stored in separate csv files. The dimension of the extracted deep or bottleneck features is 64.

4.4 Comparison of different algorithms

We compute distances based on the k nearest neighbors. i.e., we construct the vector, $D_i, i = 1, .., l$ using k nearest neighbors where $D_i = |N_i|/k$, and $N_i \in \{N(i) : C(N(i)) = C(i)\}$, where $N(i)$ and $C(i)$ represent the neighbors and the class of i respectively. Any of the suitable distance metric such as, the Euclidean distance or learned Mahalanobis metrics (for instance, LMNN [46], LFDA [40], DistBoost [21]) can be used. Fig. 6 shows the decision boundary generated by SVM and proposed *k*NN-SVM for both training and test set. To visualize the shape of decision boundaries of different SVM and *k*NN on the high dimensional (33 features) deep features, PCA is applied on deep features and retain those higher components contributing 99% of the total variances of

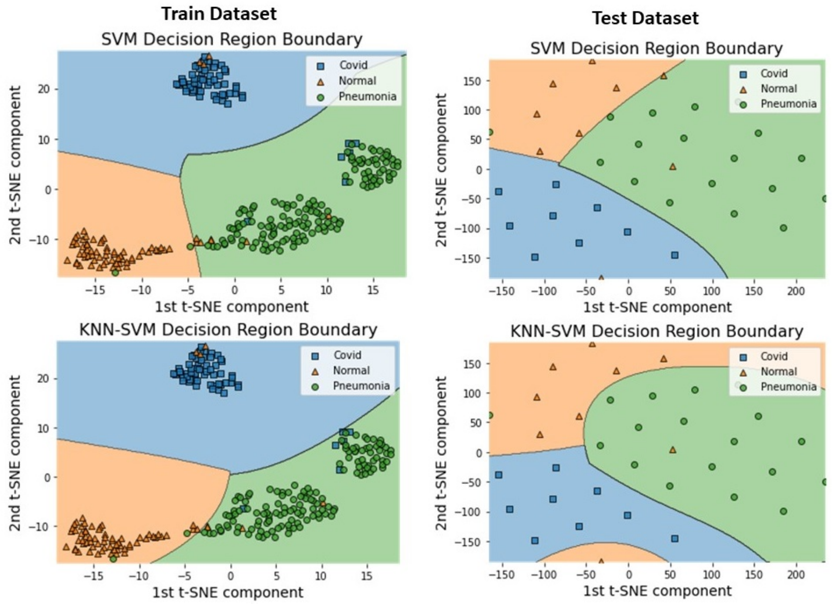


Fig. 6: Decision boundary generated by SVM and proposed kNN-SVM on training and test data separately.

the data. Then, t-SNE [19] is used to reduce data obtained from PCA to two dimensions which is illustrated in Fig. 6. From this result, it is prominent that kNN-SVM offers less misclassification error for both training and test dataset. Table 2 demonstrates the comparison among VGG16 transfer learning, SVM, SVM with clustering information, kNN-SVM, and kNN-SVM with clustering algorithms in terms of Accuracy, Precision, Recall and F1-Score. Accuracy is calculated as the fraction of correct predictions in the test dataset. Precision is the ability of the model to identify only the relevant instances. Recall is the ability of a model to find all the relevant cases. F1-score is the harmonic mean of Precision and Recall. Table 2 shows that the performance of kNN-SVM outperforms all other models. In addition, incorporating clustering information into classification tasks enhances the discriminating ability of the classifiers. Fig. 7 illustrates the Receiver Operating Characteristic (ROC) and Precision-Recall curves for all the algorithms executed in this study. The area under the ROC curve and Average precision are also mentioned in the legend of the figure. Results show that kNN-SVM with clustering information is superior to other algorithms (area under ROC and Average Precision (AP) for kNN-SVM are is 0.93 and 0.87 respectively).

Table 2: Performance Analysis of Different Models for Three Classes (COVID-19, Pneumonia, and Normal) Classification.

Algorithm	Accuracy	Precision	Recall	F1-score
VGG16 + Transfer Learning	0.75	0.73	0.69	0.66
SVM	0.86	0.88	0.83	0.85
SVM + Clustering	0.86	0.88	0.83	0.85
*k*NN-SVM	0.89	0.89	0.89	0.89
*k*NN-SVM+Clustering	**0.92**	**0.92**	**0.91**	**0.91**

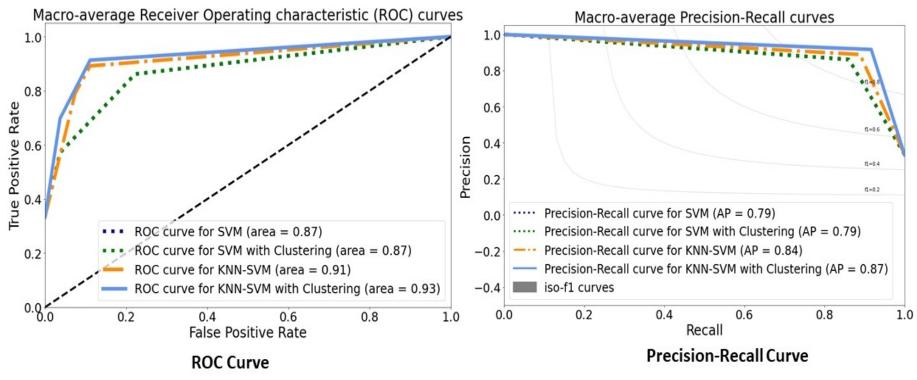


Fig. 7: ROC and Precision-Recall curve for different algorithms.

4.5 Error Analysis

To demonstrate the explainability of the deep learning models, Gradient-weighted Class Activation Map (Grad-CAM) [38] and Layer-wise Relevance Propagation (LRP) [5] algorithms are resorted to visualize class activation maps. Grad-CAM is a class-discriminative localization technique used for representing any CNN model more interpretable through visual explanations. It works by mapping the input image to the activations of the last convolution layer and the output predictions, and then computing the gradient of the top predicted class with respect to the activations of the last convolution layer for input image. Grad-CAM gives a heatmap as output, which is exploited to investigate the important areas in the images that VGG16 model emphasizes to correctly classify the images. Fig. 8 shows the individual class activation heatmap and superimposed it on five representative CXR images, out of which, three were correctly classified and two were misclassified by the VGG16 model as shown in the figure. Fig. 9 illustrates five representative images, out of them, the first three images (a-c) were misclassified by all algorithms reported in this study. The last two images (e-f) are correctly classified by the *k*NN-SVM algorithm only. We prepared Fig. 9 using the iNNvestigate library [3]. Gradient column in Fig. 9 shows the output neuron's gradient with respect to the input. Deep Taylor finds a root point for each neuron that is close to the input but has a zero output value, and then

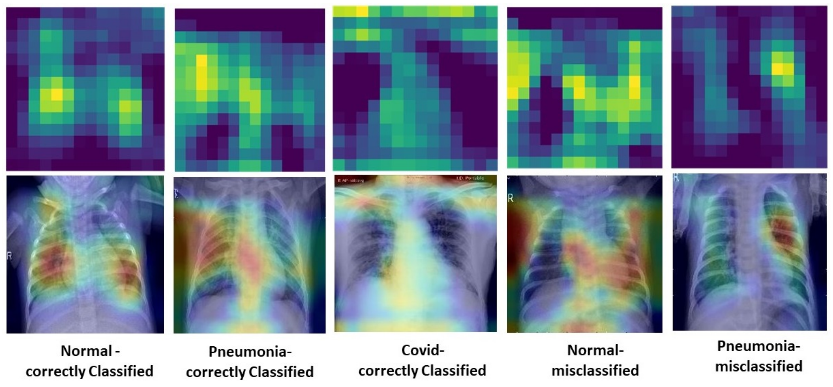

Normal - correctly Classified
Pneumonia- correctly Classified
Covid- correctly Classified
Normal- misclassified
Pneumonia- misclassified

Fig. 8: Heatmap plus class activation for different chest x-ray images.

recursively estimates each neuron's attribution using the difference [29]. LRP recursively attributes each neuron's input relevance proportionally to its contribution to neuron output. We used different LRP presets on iNNvestigate such as "LRP-X", "LRP-Epsilon", and "LRP-PresetAFlat" [5].

These five challenging images were analyzed by a physician. Image (a) is a child patient with supine positions. Image (b) belongs to a neonate patient with both lung field pneumonitis. slight right lung middle lower zone pneumonitis is found in image (c). Clustering information helps the *k*NN-SVM algorithm to detect both lung field pneumonitis present in Image (d). Image (e) depicts right lung pneumonitis, as well as emphycematous change in both lung fields. *k*NN-SVM fails to classify pneumonia from a few children or neonates patients as well as the existence of pneumonitis at a lower degree. Physicians find challenging these five images as difficult as deep learning models and thus the experimental results reported in this experiment demonstrate that the performance of all the models reported in this study is at least as comparable to radiologists. Among all of them, the proposed *k*NN-SVM with deep features outperforms better than other algorithms.

5 Conclusion and Future Works

In this research we improved the performance of the transfer learning with the VGG16 model for COVID-19 pneumonia detection through a novel *k*NN regularized Support Vector Machine (*k*NN-SVM). The *k*NN prioritizes the examples that are mostly surrounded by the examples from its own class to correctly classify while deciding decision boundary during training. This study shows that incorporating clustering information improves the performance of the classifiers. Results demonstrate that the machine learning algorithms are able to detect COVID-19 induced pneumonia from chest X-ray as accurate as radiologists. Extensive testing of the proposed method on other medical imaging classification problems remains an intriguing future direction.

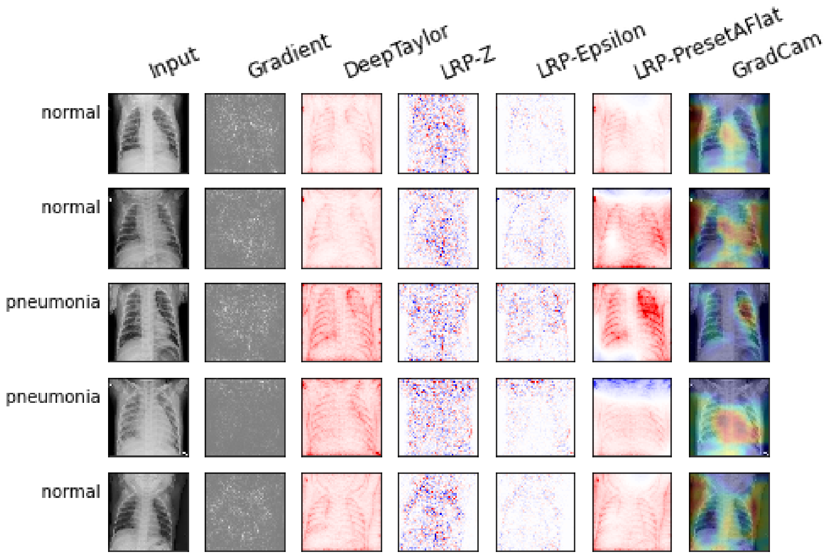


Fig. 9: Representative Chest x-ray images misclassified by several algorithms.

References

1. Ahmed, S., Yap, M.H., Tan, M., Hasan, M.K.: Reconet: Multi-level preprocessing of chest x-rays for covid-19 detection using convolutional neural networks. medRxiv (2020)
2. Al-Waisy, A.S., Al-Fahdawi, S., Mohammed, M.A., Abdulkareem, K.H., Mostafa, S.A., Maashi, M.S., Arif, M., Garcia-Zapirain, B.: Covid-chexnet: hybrid deep learning framework for identifying covid-19 virus in chest x-rays images. Soft computing pp. 1–16 (2020)
3. Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K.T., Montavon, G., Samek, W., Müller, K.R., Dähne, S., Kindermans, P.J.: innvestigate neural networks! Journal of Machine Learning Research **20**(93), 1–8 (2019), http://jmlr.org/papers/v20/18-540.html
4. Apostolopoulos, I.D., Mpesiana, T.A.: Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in Medicine **43**(2), 635–640 (2020)
5. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
6. Bassi, P.R., Attux, R.: A deep convolutional neural network for covid-19 detection using chest x-rays. Research on Biomedical Engineering pp. 1–10 (2021)
7. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Mach. Learn. Res. **7**, 2399–2434 (2006)
8. Berkhin, P.: A survey of clustering data mining techniques. In: Grouping multidimensional data, pp. 25–71. Springer (2006)

9. Bernheim, A., Mei, X., Huang, M., Yang, Y., Fayad, Z.A., Zhang, N., Diao, K., Lin, B., Zhu, X., Li, K., et al.: Chest ct findings in coronavirus disease-19 (covid-19): relationship to duration of infection. Radiology p. 200463 (2020)
10. Brunese, L., Mercaldo, F., Reginelli, A., Santone, A.: Explainable deep learning for pulmonary disease and coronavirus covid-19 detection from x-rays. Computer Methods and Programs in Biomedicine **196**, 105608 (2020)
11. Cheng, H., Tan, P.N., Jin, R.: Efficient algorithm for localized support vector machine. IEEE Trans. on Knowl. and Data Eng. **22**(4), 537–549 (Apr 2010)
12. Chollet, F., et al.: Keras. https://keras.io (2015)
13. Cohen, J.P., Morrison, P., Dao, L.: Covid-19 image data collection. arXiv 2003.11597 (2020), https://github.com/ieee8023/covid-chestxray-dataset
14. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. **13**(1), 21–27 (1967)
15. Crammer, K., Singer, Y., Cristianini, N., Shawe-taylor, J., Williamson, B.: On the alg. implemen. of multiclass kernel-based vector machines. Mach. Learn. Res. pp. 265–292 (2001)
16. Das, N.N., Kumar, N., Kaur, M., Kumar, V., Singh, D.: Automated deep transfer learning-based approach for detection of covid-19 infection in chest x-rays. Irbm (2020)
17. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., 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)
18. Hable, R.: Universal consistency of localized versions of regularized kernel methods. Journal of Machine Learning Research **14**(1), 153–186 (2013)
19. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2nd edn. (2009)
20. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification. IEEE Trans. Pattern Anal. Mach. Intell. **18**(6), 607–616 (Jun 1996). https://doi.org/10.1109/34.506411, http://dx.doi.org/10.1109/34.506411
21. Hertz, T., Bar-hillel, A., Weinshall, D.: Learning distance functions for image retrieval. In: Proceedings of the IEEE CVPR. pp. 570–577 (2004)
22. Islam, M.M., Poly, T.N., Alsinglawi, B., Lin, M.C., Hsu, M.H., Li, Y.C.J.: A state-of-the-art survey on artificial intelligence to fight covid-19. Journal of Clinical Medicine **10**(9) (2021). https://doi.org/10.3390/jcm10091961, https://www.mdpi.com/2077-0383/10/9/1961
23. Jaiswal, A.K., Tiwari, P., Rathi, V.K., Qian, J., Pandey, H.M., Albuquerque, V.H.C.: Covidpen: A novel covid-19 detection model using chest x-rays and ct scans. Medrxiv (2020)
24. Kunapuli, G., Shavlik, J.: Mirror descent for metric learning: A unified approach. In: Proc. ECML '12. pp. 859–874 (2012)
25. Ladicky, L., Torr, P.H.S.: Locally linear support vector machines. In: ICML. pp. 985–992 (2011)
26. Liu, Y.H., Chen, Y.T.: Face recognition using total margin-based adaptive fuzzy support vector machines. IEEE Transactions on Neural Networks **18**(1), 178–192 (2007)
27. Lv, D., Qi, W., Li, Y., Sun, L., Wang, Y.: A cascade network for detecting covid-19 using chest x-rays. arXiv preprint arXiv:2005.01468 (2020)
28. Mei, X., Lee, H.C., Diao, K.y., Huang, M., Lin, B., Liu, C., Xie, Z., Ma, Y., Robson, P.M., Chung, M., et al.: Artificial intelligence–enabled rapid diagnosis of patients with covid-19. Nature medicine **26**(8), 1224–1228 (2020)

29. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017). https://doi.org/https://doi.org/10.1016/j.patcog.2016.11.008, https://www.sciencedirect.com/science/article/pii/S0031320316303582
30. Mooney, P.: Kaggle dataset:chest x-ray images (pneumonia) (2017), https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
31. Nakayama, H., Yun, Y.: Generating support vector machines using multi-objective optimization and goal programming. In: Jin, Y. (ed.) Multi-Objective Machine Learning, pp. 173–198. Springer (2006)
32. Ozturk, T., Talo, M., Yildirim, E.A., Baloglu, U.B., Yildirim, O., Acharya, U.R.: Automated detection of covid-19 cases using deep neural networks with x-ray images. Computers in biology and medicine **121**, 103792 (2020)
33. Panwar, H., Gupta, P., Siddiqui, M.K., Morales-Menendez, R., Singh, V.: Application of deep learning for fast detection of covid-19 in x-rays using ncovnet. Chaos, Solitons & Fractals **138**, 109944 (2020)
34. Pham, T.D.: Classification of covid-19 chest x-rays with deep learning: new models or fine tuning? Health Information Science and Systems **9**(1), 1–11 (2021)
35. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al.: Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225 (2017)
36. Salman, F.M., Abu-Naser, S.S., Alajrami, E., Abu-Nasser, B.S., Alashqar, B.A.: Covid-19 detection using artificial intelligence (2020)
37. Segata, N., Blanzieri, E., Bottou, L.: Fast and scalable local kernel machines. J. Mach Learn Res p. 1883 (2009)
38. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
40. Sugiyama, M.: Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. J. Mach. Learn. Res. **8**, 1027–1061 (May 2007), http://dl.acm.org/citation.cfm?id=1248659.1248694
41. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters **9**(3), 293–300 (1999)
42. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: In ICML (2004)
43. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Machine Learning Research **6**, 1453–1484 (2005)
44. Waheed, A., Goyal, M., Gupta, D., Khanna, A., Al-Turjman, F., Pinheiro, P.R.: Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection. Ieee Access **8**, 91916–91923 (2020)
45. Wang, Y., Yao, H., Zhao, S.: Auto-encoder based dimensionality reduction. Neurocomputing **184**, 232–242 (2016)
46. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**, 207–244 (Jun 2009), http://dl.acm.org/citation.cfm?id=1577069.1577078

47. Xu, H., Caramanis, C., Mannor, S.: Robustness and regularization of support vector machines. J. Mach. Learn. Res. **10**, 1485–1510 (Dec 2009), http://dl.acm.org/citation.cfm?id=1577069.1755834
48. Xu, L., Schuurmans, D.: Unsupervised and semi-supervised multi-class support vector machines. In: National Conference on Artificial Intelligence. pp. 904–910. AAAI (2005), http://dl.acm.org/citation.cfm?id=1619410.1619478
49. Yoon, M., Yun, Y., Nakayama, H.: A role of total margin in support vector machines. In: Proceedings of the International Joint Conference on Neural Networks. vol. 3, pp. 2049–2053 (2003). https://doi.org/10.1109/IJCNN.2003.1223723
50. Zhang, H., Berg, A.C., Maire, M., Malik, J.: Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2. CVPR '06 (2006)
51. Zhu, J., Rosset, S., Hastie, T., Tibshirani, R.: 1-norm support vector machines. Tech. rep., Advances in Neural Information Processing Systems (2003)