# Face Mask Detection for Disease X: A Probabilistic Approach

**Fan Jue**
School of Computing
National University of Singapore
13 Computing Drive, Singapore 117417
`e0559110@u.nus.edu`

**Gao Gui**
School of Computing
National University of Singapore
13 Computing Drive, Singapore 117417
`gaogui@u.nus.edu`

**Mo Yunbin**
School of Computing
National University of Singapore
13 Computing Drive, Singapore 117417
`yunbin.mo@u.nus.edu`

**Tian Xiao**
School of Computing
National University of Singapore
13 Computing Drive, Singapore 117417
`tianxiao02@u.nus.edu`

## Abstract

Face mask is potentially an essential safety measure to prevent the transmission of Disease X in future. Currently, manual checking is the commonly adopted means to enforce mandatory mask-wearing but it incurs high costs due to its dependence on manpower. The rapid development of machine learning (ML) offers us a more efficient method to detect facial masks. However, traditional image classification techniques cannot be directly applied because the image data captured by surveillance cameras can be noisy or blurry. Moreover, we should not alert people unless we have high confidence in the prediction outcome. In this paper, we compare several architectures which can satisfy these requirements. By comparing the performances of these architectures, we find out that the architecture with a wavelet denoising with Bayes shrink followed by a Bayesian convolutional neural network is most suitable for this scenario.

## 1 Introduction

In February 2018, World Health Organization (WHO) raised the concept of *Disease X* which refers to any unknown pathogen that can cause global epidemics in future [18]. Many countries have realized and emphasized the importance of precautionary and emergency measures against Disease X because of COVID-19, arguably the first Disease X [5, 6]. Particularly, we need to make improvements on our areas of ineffectiveness and inefficiency in combatting COVID-19 to better prepare us for future Disease X.

Similar to many influenza, airborne droplet is one of the major means of transmission of COVID-19. Therefore, mandatory indoor mask-wearing has been widely adopted by countries and organizations in recent years. To enforce this safety measure, a huge amount of manpower is put into checking face masks. However, this system relies excessively on manual labour and incurs extra surveillance fees. Moreover, the efficiency of manual checking is limited when the crowd far outnumbers personnel in charge. Therefore, it is essential to devise an efficient alternative system to achieve the same purpose, potentially through images captured by closed-circuit television (CCTV) cameras. Conceptually, our system needs to automatically differentiate those who wear and do not wear masks based on the captured images.

In recent years, machine learning (ML) techniques have been rapidly evolving and extensively used to automate data-driven tasks, including image classification tasks. The fundamental idea of (supervised) ML is to train a model to represent the underlying relationships between data and their labels. There have been existing techniques such as convolutional neural networks (CNNs) designed for basic image classification tasks and empirically shown to perform well [13]. However, we cannot directly utilize such techniques in our system due to several additional requirements arising from our problem setting. Firstly, the captured images may be noisy or blurry depending on the condition of cameras and environments. Secondly, we should not force our system to make a decision when it is uncertain about whether a person is wearing a mask, since we do not want to alert or penalize those who are actually wearing masks properly. Thereby, our system must be able to represent or quantify the uncertainty of its decision. In summary, **the aim of our project is to build an image classification system for mask detection that can satisfy both aforementioned requirements.**

## 2 Related Works

Our work is closely related to denoising representation learning for images and image classification with uncertainty quantification. The subsections below introduce related works in these two areas.

### 2.1 Denoising Representation Learning for Images

Image denoising and representation learning play important roles in many image processing applications such as image classification and image segmentation. Image denoising is the process of correcting or recovering noisy or blurry images. Typically, models based on Markov random fields [24] are used to model influences between pairs of pixels by leveraging the fact that original images are usually smooth. Using such techniques, one can exploit contextual information from the neighbours of a pixel to identify and correct the noise. Meanwhile, representation learning of images aims to get a compact representation for downstream tasks. The autoencoder network structure is one commonly used unsupervised learning technique which applies non-linear transformation on original representation. Some variants of autoencoders have been proven effective in representation learning. For example, richer spatial features can be captured by more complex multi-level variational autoencoders (VAEs) [22] and ensemble techniques can be used to boost performance [1].

Recent works [8, 14] have successfully integrated image denoising with representation learning to directly learn a denoised representation. Our project is based on these techniques for the processing of noisy or blurry images.

### 2.2 Image Classification with Uncertainty

Image classification has been an important field of study in ML. Since the inputs are graphical in image classification, they always have a large number of features, leading to the well-known "curse of dimensionality" problem [10]. Conventional image classification techniques address this problem by convolving and pooling raw inputs in order to reduce dimensionality while preserving relationships in spatial locality. In this way, an *embedding* of the original image is created. Basic ML models such as fully connected neural networks [12], decision trees [11] or support vector machines [21] can then be applied to the embeddings for the classification task. Although these techniques have been widely used in industry for their satisfactory performance and efficiency, they cannot model or quantify the uncertainty in their predictions. For this reason, recent research has started combining image convolutions with probabilistic models such as Naïve Bayes [2]. With the development of *Bayes by Backprop* [3] which integrates Bayes' theorem into backpropagation, Bayesian convolutional neural networks (BCNNs) [17] have also been developed in order to model the distribution of weights at each layer on top of basic CNNs.

Another family of non-linear and non-parametric probabilistic models is the Gaussian process (GP) models [16], which model the underlying distribution of functions and produce probabilistic insights by nature. Although the prohibitive computational costs have rendered GP unfit for high-dimensional inputs like images, GP provides unique strength in automatically tuning hyperparameters. By incorporating convolution into GP, convolutional Gaussian processes (CGPs) [19] maintain the strength of GP and can be applied to image classification tasks.

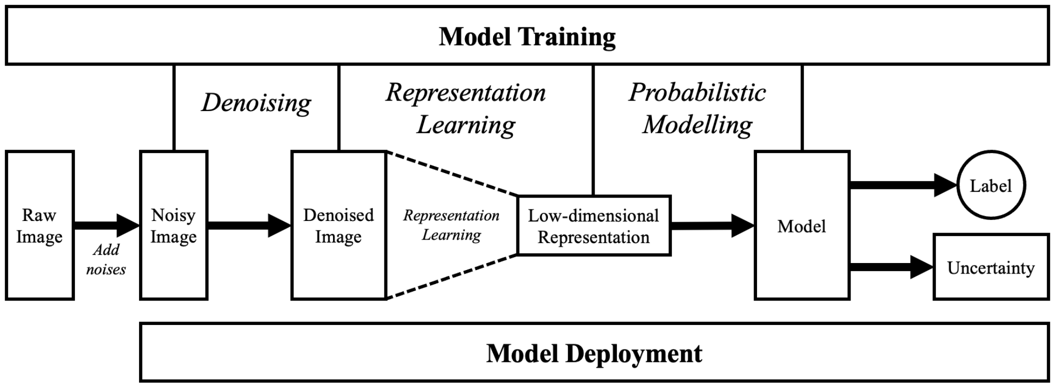


Figure 1: General framework of face mask detection system.

## 3 Methodology

The general framework of our proposed system is shown in Figure 1. Based on our problem setting, there are three main components in our framework: denoising, representation learning and probabilistic modelling. Although there are many existing techniques that can handle each individual component, a totally discrete training process is not expected to perform well since each component serves different purposes and errors will accumulate throughout the separate processes. However, it is also infeasible to pipeline the three components into a single end-to-end training process because they are intended to optimise different objective functions. Thus, as a surrogate measure, we decide to combine the training processes of as many components as possible. We identify two possible realizations: (1) denoising representation learning (DRL) + probabilistic modelling, and (2) denoising + convolutional probabilistic modelling. Therefore, our whole methodology is as follows: In order to build a noise-robust system, we first add Gaussian noises to each graphical input to create a noisy image. We then conduct two lines of experiments:

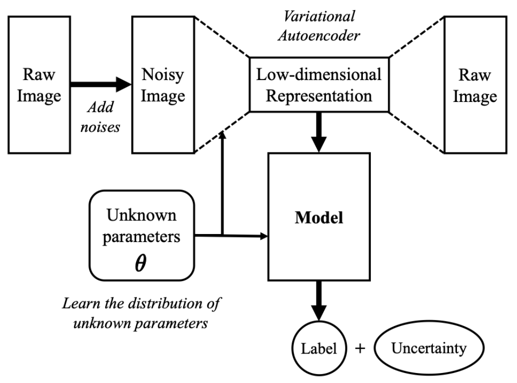
- In the first line, we perform denoising representation learning to obtain a low-dimensional **denoised** representation of the noisy image in order to make our system more robust towards potentially noisy or blurry images captured by CCTV cameras. Then we feed the feature representation data to probabilistic models to learn the distribution of unknown parameters. In the model deployment stage, representations of actual images are fed to the model to obtain a probability distribution over all possible outcomes.
- In the second line, we first denoise the image using a probabilistic approach. Then we feed the denoised images into convolutional probabilistic models for both representation learning and model training with uncertainty quantification. In the model deployment stage, actual images are denoised and then fed to the model.

Specific techniques used for both lines are illustrated in detail in Sections 3.1 (Figure 2a) and 3.2 (Figure 2b). We will compare the results given by various implementations in both lines and choose the best one as our final system.

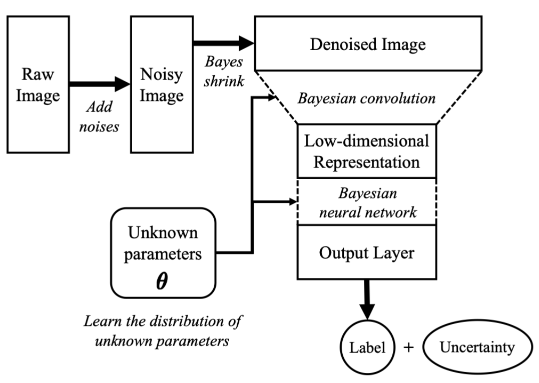
### 3.1 Denoised Representation Learning (DRL) and Probabilistic Modelling

#### 3.1.1 DRL

To reduce the dimensionality of input images for downstream training tasks, we can instead use a latent embedding learnt by an autoencoder. However, conventional autoencoders do not have regularization over the latent space. Given the aim to minimize reconstruction loss, the autoencoder could exploit the possibilities of overfitting to achieve optimal encoding with information loss. To mitigate the problem, we introduce variational autoencoder (VAE) as a probabilistic treatment at this step. VAE is used to learn smooth latent state representations of the input data. The inputs are encoded as a distribution over the latent space so that information encoded in the latent space does not become completely detached from each other. Instead, they should be more "continuous" in the sense that two close points should give similar results once decoded.



(a) Denoised representation learning and probabilistic modelling.



(b) Probabilistic denoising and convolutional probabilistic modelling.

Figure 2: Two main architectures in our project.

**Denoising Variational Autoencoders (DVAEs).** In order to make our system robust towards noises, we use a variant of VAE called denoising variation autoencoder (DVAE). Hypothetically, partially destroyed input should yield almost the same representation as a perfect one, because a good representation is expected to capture stable structures in the form of dependency and regularity characteristics of the (unknown) distribution of its observed input. Referring to preceding studies [14], we integrate a noise-adding process into a classical VAE to improve the robustness of our model. In the noise-adding process, we first make a corrupted version of the original image by adding noises which follow a known distribution. The network then learns to reconstruct the uncorrupted original input [8]. Compared to regular VAE, DVAE models the posterior probability of latent representation given the input using a mixture of Gaussian instead of a single Gaussian, which increases the robustness of the representation as the encoder also learns how to map corrupted input to the true distribution.

**Dual-Directional Strategies.** Inspired by previous work [20], we additionally adopt a dual-direction strategy to preserve spectral-spatial coherence. Specifically, the spatial continuity is captured by flattening neighbouring samples in dual classic directions; secondly, the spectral–spatial continuities are captured in the forward and backward directions. This strategy provides a feasible way to avoid the loss of spatial information when flattening samples in the spatial domain, without using any convolutional layers.

### 3.1.2 Probabilistic Modelling

Through DRL, we obtain a denoised and compressed representation of the raw images. We then apply various probabilistic modelling techniques to the representation in order to learn the relationship between graphical features and labels with uncertainty. Specifically, we use a set of simple probabilistic models such as Naïve Bayes classifier and Bayesian logistic regression since the input dimension is very low at this stage and there is no need to use any complex model.

## 3.2 Probabilistic Denoising and Convolutional Probabilistic Modelling

### 3.2.1 Wavelet Thresholding with Bayes Shrink

Images can be transformed into waves using wavelet transform [23], and waves can be characterized by a set of *wavelet coefficients*. Waves given by images without noises are usually patterned and smooth and the wavelet coefficients are large. On the other hand, noises lead to chaotic and erratic waves whose wavelet coefficients are small. Therefore, a common technique for image denoising is wavelet thresholding, where we set a threshold and "shrink" wavelet coefficients that are smaller than the threshold.

A Bayes shrink [4] is an adaptive way to find the appropriate threshold with Bayes' theorem. Specifically, the total loss can be estimated by a Bayes estimator with generalized Gaussian distribution as the prior. By minimising this loss, we can find the appropriate threshold to shrink the noises.

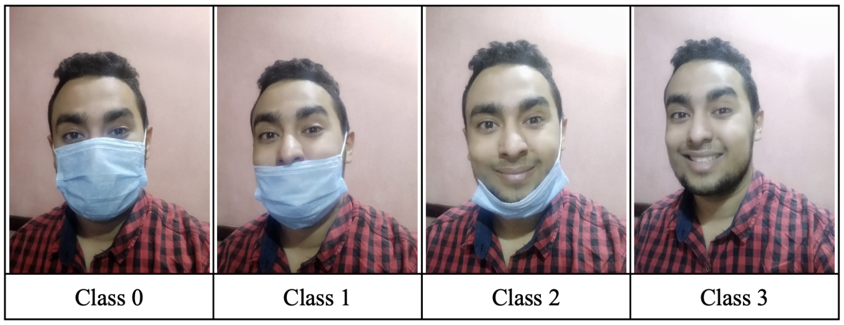


Figure 3: Examples of training images and their labels.

### 3.2.2 Convolutional Probabilistic Modelling

Convolution is a prevalent dimensionality reduction technique that works well for images. The basic idea of convolution is to perform an element-wise multiplication between tensors and a sliding convolutional kernel. Hence, each kernel can be considered as a set of weights assigned to each data, and we can model the weights as a probabilistic distribution using Bayesian techniques. Specifically, we use the following two convolutional probabilistic models:

**Bayesian Convolutional Neural Networks (BCNN).** Similar to a CNN, a BCNN [17] consists of a set of convolutional and pooling layers, followed by a fully connected perceptron neural network. However, instead of obtaining deterministic weight matrices at each convolutional and fully-connected layer through backpropagation, a BCNN focuses on modelling the distribution of weights. To integrate Bayes' theorem into backpropagation, Blundell *et al.* [3] developed the *Bayes by Backprop* algorithm which performs variational approximation of the posterior distributions at each layer through Monte Carlo gradient estimation. This algorithm allows us to model the uncertainty in every parameter in the neural network and hence the uncertainty in final predictions.

**Convolutional Gaussian Process (CGP).** Instead of conventional convolutional layers used in CNNs, a CGP [19] repeatedly uses a non-linear and non-parametric patch response function as filters, where the patch response function is assumed to follow a Gaussian distribution of functions. The overall function for the convolutional kernel then takes the sum of all patch responses and hence also follows a Gaussian distribution of functions. As such, the Gaussian process itself is convolved, saving the need to introduce additional parameters as compared to using CNNs. Since direct computation of the true posterior is intractable, a variational framework is adopted to approximate posterior using a small set of inducing points. The inter-domain covariances in the approximation can be easily found via the construction of convolutional kernel.

## 4 Experiments and Results

### 4.1 Data

In this project, we use images from Kaggle's Face Mask Detection dataset [15] as inputs. It has 20,000 images comprising 5,000 unique faces. As shown in Figure 3, each image is labelled as 0 (mask covers both mouth and nose), 1 (mask covers mouth but not nose), 2 (mask does not cover mouth or nose), or 3 (no mask). In our experiments, we used 2,000 out of the 20,000 images for training due to limited compute resource. We first resize the coloured input images to $256 \times 256$ pixels and then process them into gray-scale images where the intensity values are normalized such that each pixel value ranges between $[0, 1]$. To create noisy images, we add a Gaussian noise of $\mathcal{N}(0, 0.025^2)$ to each normalized image. Figures 4a and 4b show a gray-scale processed image and a synthesized noisy image respectively.

### 4.2 Denoised Representation Learning and Probabilistic Modelling

We consider fully connected and convolutional neural network architectures for VAEs with denoising and dual-directional strategy respectively. Our architecture is shown in Figure 9a and **??** in the appendix. For both architectures, we use a single layer of size 64 for the latent variables. In the fully connected architecture, the encoder consists of an input and a hidden layer, each with 256

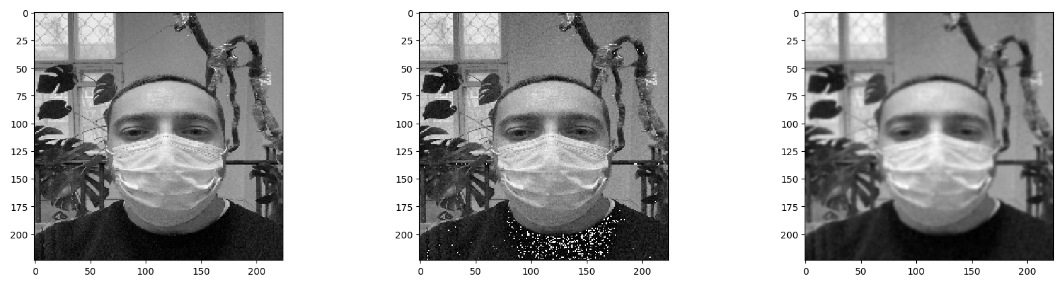

(a) Original image. (b) Noisy image. (c) Denoised image.

Figure 4: Illustration of noise-adding and denoising process.

units, and the decoder consists of a hidden and an output layer, each with 256 units. For the CNN architecture, we apply a $4 \times 4$ kernel on the input and extract 32 features, which is followed by a hidden convolutional layer with 64 output channels and again a $4 \times 4$ kernel. And the decoder has a similar but reversed layer structure to produce the 1-channel reconstructed image. The last layer of both architectures is sigmoid and we use a combination of Binary Cross Entropy and KL divergence as the loss term. For both VAE variants, we train and optimize all models using Adam [9] for 10 epochs by setting the batch size to 32, and the learning rate to 0.0001.

For denoising VAE, we inject noise at the input level with standard normal distribution multiplied by a rescaling constant and maximize the denoising variational lower bound [8]. Theoretically, we should get better performances than vanilla VAE. However, in our case, we did not observe significant differences in performance between VAE and DVAE.

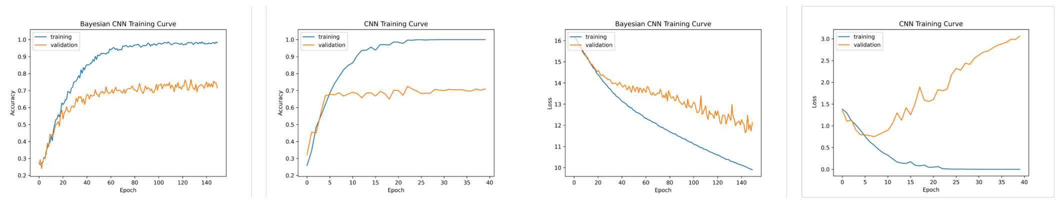
On the other hand, we also implement dual-directional VAE, which acquires a strong ability in preserving spatial and spectral relations without using convolutional layers. To compare with DVAE, we follow the same setup. The major difference is that there are two images, with a pair of orthogonal directions as input to the encoder, each with an independent hidden layer. Based on the research results of the performance of different directions [20], we choose 0 degree and 90 degrees as the pair of orthogonal directions for input images, as it achieves one of the best performances. During the training process, we notice that dual-directional VAE achieves a faster convergence and a better performance compared with DAVE and VAE.

The low-dimension representation of images is then fed into Naïve Bayes and Bayesian logistic regression models for training. However, both models produce very poor results (around 0.33 training and 0.28 testing accuracies) for all VAE and DVAE variants. Moreover, even though Naïve Bayes classifier is a probabilistic model, it does not provide quantification for epistemic uncertainty, which also makes it unsuitable for our objective.

### 4.3 Probabilistic Denoising and Convolutional Probabilistic Modelling

In order to denoise the images, we apply a mild Gaussian filter followed by wavelet denoising with Bayes shrink as described in Section 3.2.1. The denoised images look like Figure 4c. We then start to tune our BCNN architecture based on the denoised images and our final architecture is shown in Figure 9c in the appendix. Specifically, for each `Conv2DReparametrization` layer and `DenseReparametrization` layer, we find out that multivariate Gaussian distribution is optimal to be used as the prior and Monte-Carlo sampling is used to approximate the posterior.

Since the architecture can be easily converted to a normal CNN by replacing `Conv2DReparametrization` and `DenseReparametrization` layers with `Conv2D` and `Dense` layers, we train a BCNN concurrently with a CNN and compare their performances and efficiency in order to evaluate the practicality of BCNN in real life. The training and validation curve is shown in Figure 5. It can be seen that both BCNN and CNN are able to attain the same training and validation accuracy, but BCNN takes around 150 epochs to reach the optimum while CNN starts to overfit after around 7 epochs because BCNN requires more model parameters and additional Monte-Carlo sampling as compared with CNN. However, what makes BCNN particularly suitable for our system is its ability to quantify both aleatoric and epistemic uncertainty. Figure 6 is a graphical illustration.



(a) (b) (c) (d)

Figure 5: Change in training and validation accuracy (a) and loss (c) as the number of epochs increases for BCNN those (b) (d) for CNN.

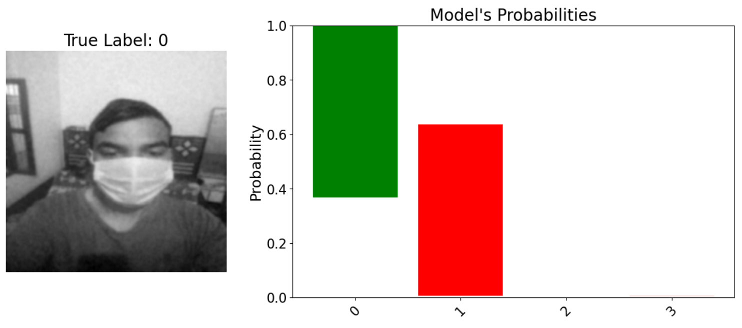


Figure 6: Uncertainty quantification given by BCNN. The coloured bar shows a $95\%$ confidence interval for model's predicted probabilities for each label $0$, $1$, $2$ and $3$.

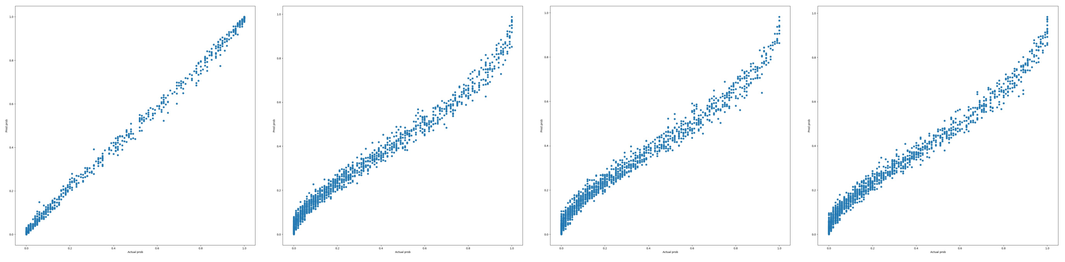
The length of each confidence interval (coloured bar) represents model's uncertainty towards the prediction it makes. Therefore, in our problem setting, we are able to identify the instances that our model is certain about.

Besides BCNN, we also attempt to use CGP as an alternative convolutional probabilistic model. Unfortunately, CGP causes the kernel to die when trained on 50 examples of shape $16 \times 16$. Also, the training for CGP is very time costly even for data of shape $8 \times 8$, and the prediction accuracy is extremely low (indeed this may be caused by the low dimension of data). Thus we conclude that even though previous research [19] has shown that convolution allows CGP to handle higher dimensional data, CGP is not suitable for our task because our images are too complicated and large for CGP to handle.

### 4.4 Comparison of Results

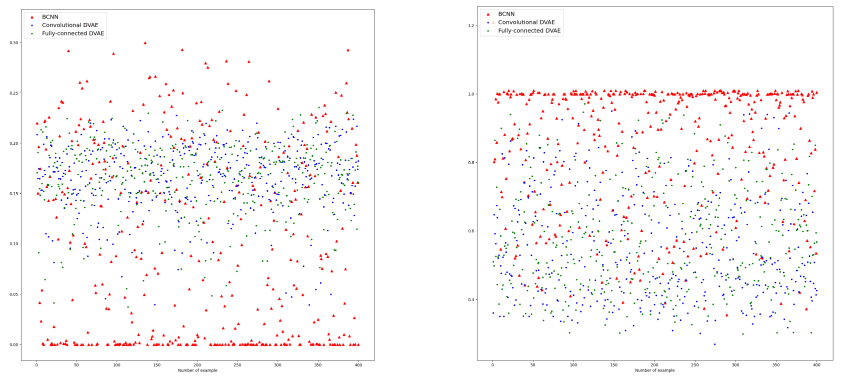
There is a vast difference in model performance between the two main architectures. The prediction accuracy and F1-score of BCNN are both around $75\%$ while those of autoencoder-based classifiers are around $25\%$, which is almost the same as a random classifier. The confusion matrices of predictions given by all architectures are given in Figure 10 in the appendix. To understand the performance in uncertainty quantification, we plot the reliability diagram [7] for these models using calibration (Figure 7). Reliability diagram plots the graph of model's predicted probability of each label against the empirical frequency that each label occurs. In the ideal case, these two should be equal and the graph is $y = x$. From Figure 7, the graph for BCNN is closest to the line $y = x$ and hence is the most reliable.

To further analyze the relationship between model uncertainty and model performance for each architecture, we plot two scatter graphs of model uncertainty and predicted probability for each testing example (Figure 8). Specifically, model uncertainty is approximated using a Monte-Carlo sampling approach with 100 random samples of model prediction probabilities. A lower model uncertainty implies that the model tends to make the same decision for multiple iterations. As shown in Figure 8a, BCNN in general gives much lower model uncertainty scores, indicating that it has lower epistemic uncertainty. On the other hand, Figure 8a demonstrates that BCNN gives the highest predicted probability. These results are coherent with our reliability diagrams, hence we conclude that BCNN is indeed the best approach to solve our problem.



(a) BCNN. (b) Convolutional DVAE. (c) Fully-connected DVAE. (d) Dual-directional VAE.

Figure 7: Reliability diagrams for two main architectures in our project.



(a) Scatter plot of model uncertainty for each testing example.

(b) Scatter plot of model's predicted probability for each testing example.

Figure 8: Comparison of performances of two main architectures in our project.

The poor performance given by DVAE might attribute to the difference in learning objectives of a VAE and a classifier. In particular, a VAE aims to ensure continuity such that two look-alike images should be close to each other in the latent space, while a classifier focuses more on distinguishing features of members of each class, which is discouraged in VAE. Also, as the CNN architecture is computationally intensive, we have to reduce the network complexity, which could potentially result in information loss. And with such a relatively small training set, it is difficult for the network to generalize well on unseen data. Learning representation with uncertainty may potentially be an effective approach but how to improve its effectiveness for downstream classification tasks needs exploration.

## 5 Conclusion

In conclusion, our final proposed image classification system consists of wavelet denoising with Bayes shrink followed by a BCNN for probabilistic image classification. We empirically demonstrate that after probabilistic denoising, BCNN can achieve the same training and testing accuracy as CNN, and it can also provide competitive uncertainty quantification. Therefore, this system can satisfy the desirable requirements to classify face masks with a confidence level.

## References

[1] Daniel Addo et al. “EVAE-Net: An Ensemble Variational Autoencoder Deep Learning Network for COVID-19 Classification Based on Chest X-ray Images”. In: *Diagnostics* 12.11 (2022), p. 2569.

[2] Sayali Ambekar and Rashmi Phalnikar. “Disease risk prediction by using convolutional neural network”. In: *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE. 2018, pp. 1–5.

[3] Charles Blundell et al. “Weight uncertainty in neural network”. In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[4] S Grace Chang, Bin Yu, and Martin Vetterli. “Adaptive wavelet thresholding for image denoising and compression”. In: *IEEE transactions on image processing* 9.9 (2000), pp. 1532–1546.

[5] Peter Daszak. “We knew disease X was coming. It’s here now”. In: *New York Times* 27 (2020).

[6] Jason Gale. *Coronavirus may be ‘Disease X’Health Experts Warned About. Bloomberg News*. 2020.

[7] Holly C Hartmann et al. “Confidence builders: Evaluating seasonal climate forecasts from user perspectives”. In: *Bulletin of the American Meteorological Society* 83.5 (2002), pp. 683–698.

[8] Daniel Im Im et al. “Denoising criterion for variational auto-encoding framework”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.

[9] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[10] Mario Köppen. “The curse of dimensionality”. In: *5th online world conference on soft computing in industrial applications (WSC5)*. Vol. 1. 2000, pp. 4–8.

[11] Dmitry Laptev and Joachim M Buhmann. “Convolutional decision trees for feature learning and segmentation”. In: *Pattern Recognition: 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings 36*. Springer. 2014, pp. 95–106.

[12] Zewen Li et al. “A survey of convolutional neural networks: analysis, applications, and prospects”. In: *IEEE transactions on neural networks and learning systems* (2021).

[13] Keiron O’Shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).

[14] Yoshua Bengio Pascal Vincent Hugo Larochelle. “Extracting and Composing Robust Features with Denoising Autoencoders”. In: *ICML 08: Proceedings of the 25th international conference on Machine learning: Pages 1096–1103)* (2008).

[15] Kucev Roman. *500 GB of images for Face Mask Detection. Part 1*. Kaggle, 2021. URL: `https://www.kaggle.com/datasets/tapakah68/medical-masks-part1`.

[16] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. “A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions”. In: *Journal of Mathematical Psychology* 85 (2018), pp. 1–16.

[17] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. “A comprehensive guide to bayesian convolutional neural network with variational inference”. In: *arXiv preprint arXiv:1901.02731* (2019).

[18] Saurabh Shrivastava and Prateek Shrivastava. “World Health Organization releases the list of blueprint priority diseases”. In: *Journal of the Scientific Society* 45.1 (2018), pp. 49–49.

[19] Mark Van der Wilk, Carl Edward Rasmussen, and James Hensman. “Convolutional gaussian processes”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[20] He Huang Wenbo Yu and Gangxiang Shen. “Multilevel Dual-Direction Modifying Variational Autoencoders for Hyperspectral Feature Extraction”. In: *IEEE Geoscience and Remote Sensing Letters ( Volume: 19)* (2022).

[21] Wei-Chang Yeh. “Convolutional Support Vector Machine”. In: *arXiv preprint arXiv:2002.07221* (2020).

[22] Wenbo Yu, He Huang, and Gangxiang Shen. “Multilevel dual-direction modifying variational autoencoders for hyperspectral feature extraction”. In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022), pp. 1–5.

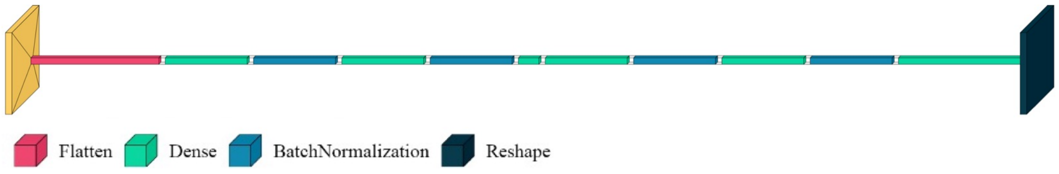
[23] Dengsheng Zhang and Dengsheng Zhang. “Wavelet transform”. In: *Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval* (2019), pp. 35–44.

[24] Ruoqiao Zhang et al. “Gaussian mixture Markov random field for image denoising and reconstruction”. In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE. 2013, pp. 1089–1092.

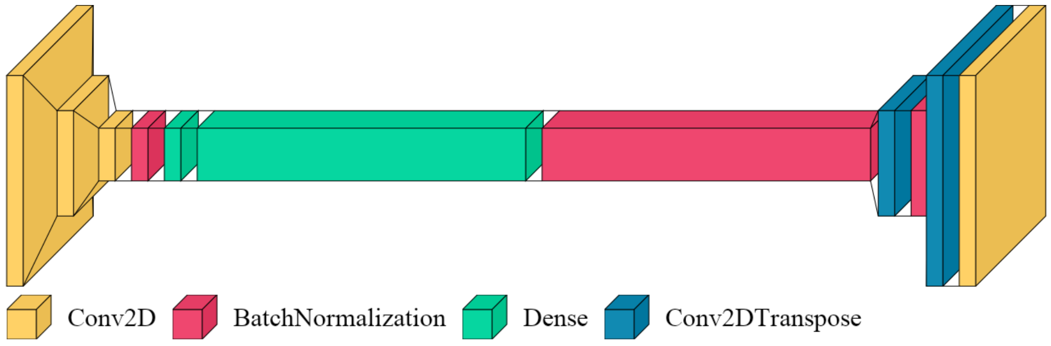
## Appendix

### Architectures of VAEs and BCNNs

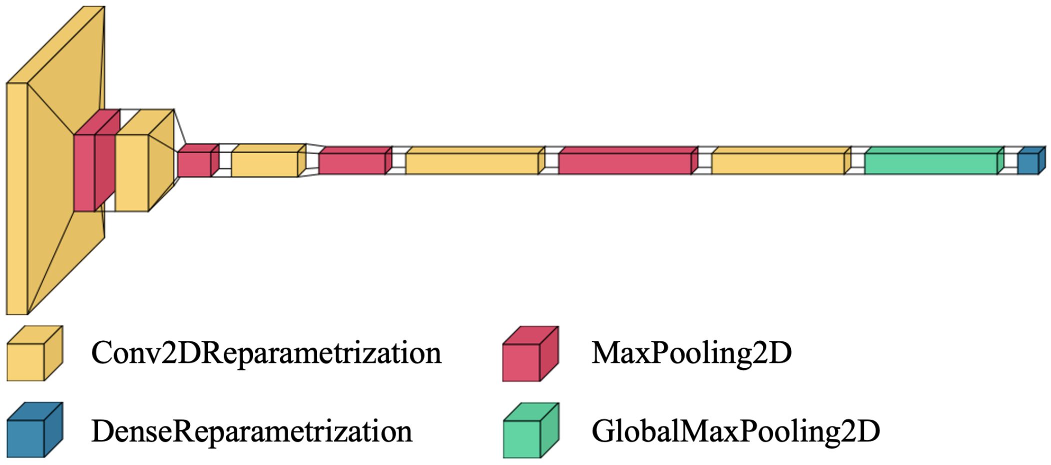
Below are the architectures of various VAEs and BCNNs that we use in this project.



(a) Fully-connected DVAE.



(b) Convolutional DVAE.

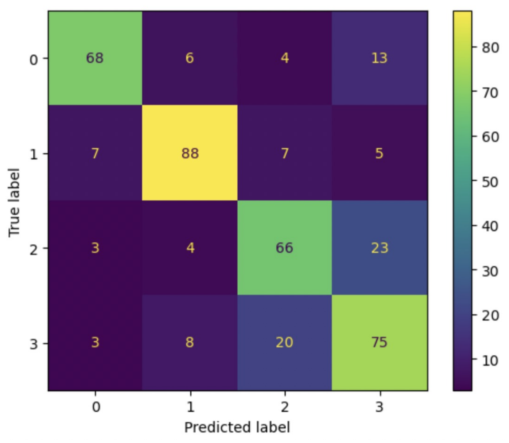


(c) BCNN.

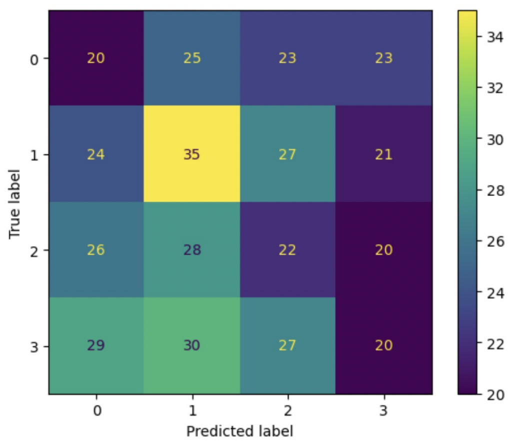
Figure 9: Architectures of VAEs and BCNNs.

## More Experimental Results

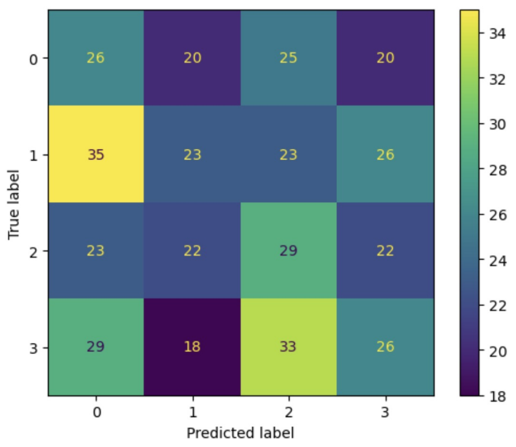
Figure 10 shows the confusion matrices of predictions given by BCNN, fully-connected DVAE, convolutional DVAE and dual-directional VAE. It is clear that BCNN has a much better performance than the others in our project.



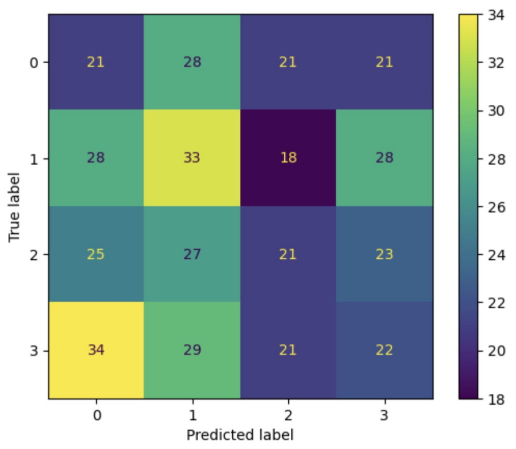
(a) BCNN.



(b) Fully-connected DVAE.



(c) Convolutional DVAE.



(d) Dual-directional VAE.

Figure 10: Confusion matrices.

## Links to Code

- `https://drive.google.com/drive/folders/13hscu63qVs9-YQx89TPXURZUfcJg5zSy?usp=share_link` (VAE).
  Referenced from `https://github.com/jiwoongim/DVAE-Pytorch-`.
- `https://drive.google.com/drive/folders/1-314ewEcKf0125Esh7a7NnrMH5qMrThp?usp=sharing` (BCNN, evalution).