Image Classification by Convolutional Neural Network with Two-stage Training

Presenter: Yi Zhou

Advisor: Professor Qiang Wu

Outline

- Problem Definition and Significance
- Convolutional Neural Network
- Two-stage Training Method
- Applications
- Results and Analysis
- Conclusions

Problem Definition and Significance

- **Definition of Computer Vision (CV):** a field of AI enabling computers to capture and comprehend information from images, videos, and other inputs.
- **Significance of CV:** Crucial for various domains, like object detection, image recognition, image classification, etc. Its applications are from healthcare, automotive to entertainment. Reduce human efforts in tasks, solve complex challenges, and improve efficiency and innovation. For example, better computer vision in video games can increase the player’s immersive experience.
- **Definition of Image Classification:** a fundamental task of computer vision that assigns labels to images based on their content.
- **Significance of Image Classification:** Crucial for various applications, including face recognition, self-driving cars, medical imaging diagnosis, etc. For example, the accuracy improvement of image classification in self-driving cars can help the car better understand the traffic scene, including traffic lights, road signs, and markings, so it can navigate more safely. In the field of medical imaging applications, image classification can better help identify the diseases for some diseases that require X-rays, CT scans, and so on.

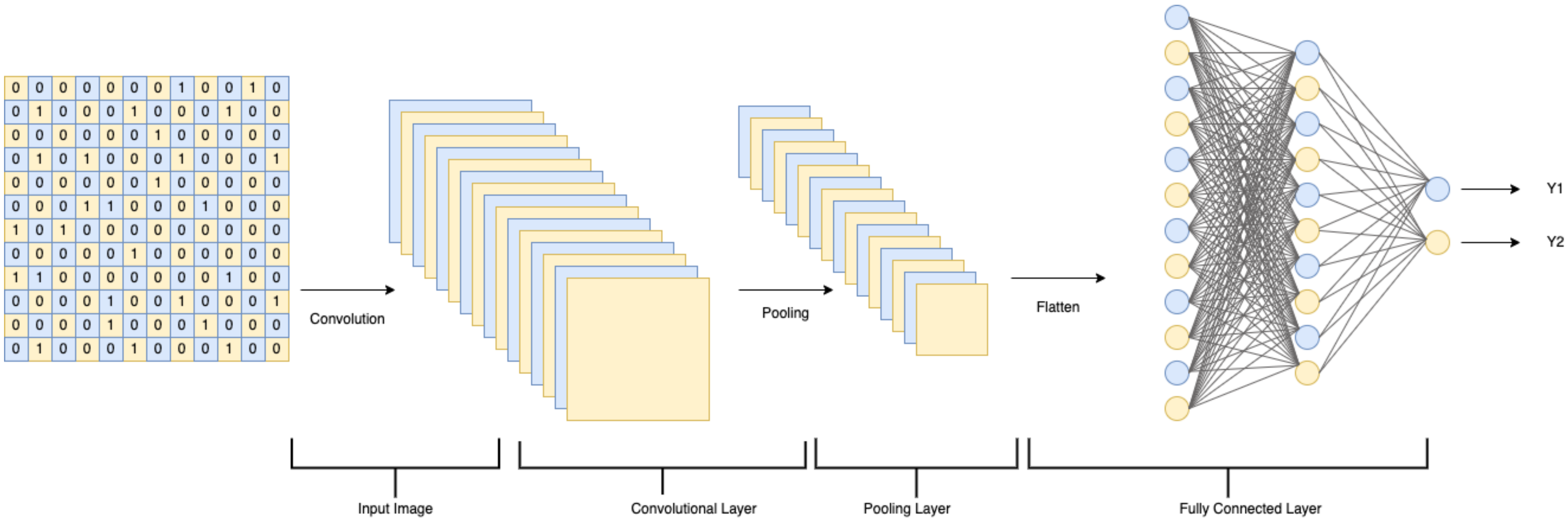
Problem Definition and Significance

- **Definition of Convolutional neural network (CNN)**: a network architecture of deep learning that is widely used in image classification, which automatically extracts features from images.
- **Significance of CNN:** No human supervision is required, and fewer parameters are trained which minimizes the computation, and overperforms traditional machine learning methods.
- **CNN architectures:** LeNet-5, AlexNet, VGG, GoogLeNet, ResNet, Xception, etc.

Problem Definition and Significance

- **Robustness of CNN model:** the ability to maintain prediction accuracy under various challenging conditions. For example, when there is noise in the data, the data quality is low, the amount of dataset is small, or the distribution of data classes is imbalanced, and so on. These can affect the robustness of the CNN.
- **Significance**: It is crucial to study the robustness and prediction accuracy of CNN models for the task of image classification. For example, during the epidemic, maintaining the robustness of the model and improving face mask image recognition’s accuracy can reduce the spread of COVID-19.

Convolutional Neural Network - Structure

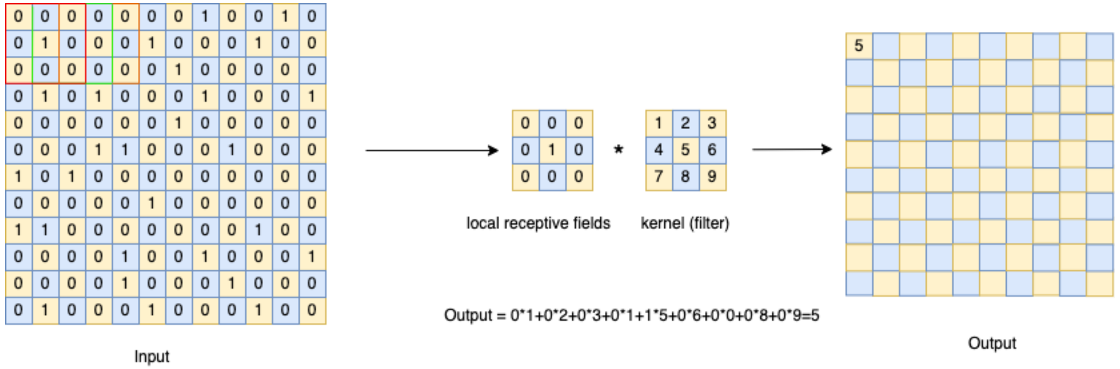


0	0	0	0	0	0	0	1	0	0	1	0
0	1	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0

Figure 1. The network structure of convolutional neural network

Convolutional Neural Network – Convolutional Layer

- Convolutional layer: sequentially convolve the local receptive field from the left top to the right button in the input image according to the convolution kernel (filter) matrix which is used to extract features from the images.



Input

0	0	0	0	0	0	0	1	0	0	1	0
0	1	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0

local receptive fields

0	0	0
0	1	0
0	0	0

kernel (filter)

1	2	3
4	5	6
7	8	9

Output = 0*1+0*2+0*3+0*1+1*5+0*6+0*0+0*8+0*9=5

Figure 2. The operation of single depth slice in a convolutional layer

output size = *(W – F* + *2P) / S* + 1

--*W*× *W*: input size

-- *F*× *F*: kernel(filter) size

--*S*: stride (the horizontal and vertical local receptive field movement steps)

--*P*: the amount of zero-padding to add 0 to the border of the input image.

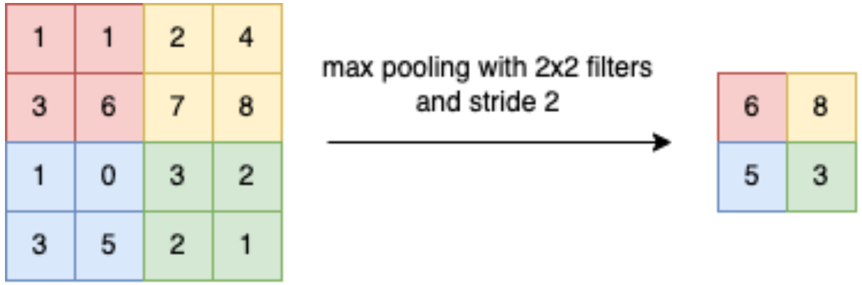
Convolutional Neural Network - Convolutional Layer

- Nonlinear activation function: Sigmoid, Tanh, ReLU, etc.

$$\begin{cases} \text{Sigmoid: } f\left(x\right) = \dfrac{1}{1+e^{-x}} \\ \text{Tanh: } f\left(x\right) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \\ \text{ReLU: } f\left(x\right) = \max\{0, x\} \end{cases} \qquad (1)$$

Convolutional Neural Network – Pooling Layer

- Pooling layer: reduce the spatial dimensions (width and height) of the feature maps from the convolutional layer, but the depth is kept the same.
- Max Pooling, Average Pooling.



1	1	2	4
3	6	7	8
1	0	3	2
3	5	2	1

6	8
5	3

Figure 3. The Max Pooling operation of single depth slice

Convolutional Neural Network - Fully connected layer

- Fully connected layer: convert the high-level feature maps from the final convolutional layer or pooling layer into category outputs.
- Nonlinear activation functions: ReLU and SoftMax.
- SoftMax: convert the raw output values from the last dense layer to category probabilities.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{N} e^{y_j}} \qquad (2)$$

-- y_i: the predicted value of class i.

Convolutional Neural Network – Regularization layers

- Batch normalization layer: a standard technique that can be applied in forward and backward propagation due to its differentiability, and it can help address the “internal covariate shift” problem.
- Dropout layer: a simple and effective regularization method for convolutional neural networks to reduce data overfitting by randomly discarding a portion of the neurons by a set probability.

Convolutional Neural Network – Model Compilation

- Model compilation: an operation after defining the architecture of convolutional neural networks. It is an important step to determine how the model is trained, optimized, and evaluated. Includes defining the loss function, optimizer, and metrics.
- Loss function: measure the performance of the model through the difference between the predicted output and the true labels. For example, categorical cross-entropy loss, sparse categorical cross-entropy loss, binary cross-entropy loss.
- Categorical cross-entropy loss:

$$CE = -\sum_i^C T_i \cdot \log\left(S(y_i)\right) \qquad (3)$$

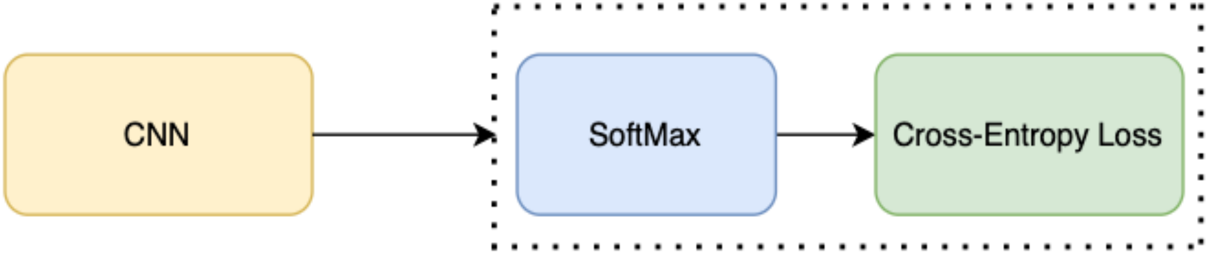
$$CE = -\log\left(S(y_p)\right) \qquad (4)$$

-- C: total number of classes

-- T: the set of targets (true label)

-- S: the softmax output

-- y_p: the positive class



Convolutional Neural Network – Model Compilation

- Optimizer: determine how to update the weights of the network during training the model to minimize the loss function. For example, Adam, SGD with momentum, RMSprop, etc.
- Adam optimizer:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \qquad (5)$$

Compute bias-corrected first moment estimate: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ (6)

Compute bias-corrected second raw moment estimate: $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ (7)

Update biased first moment estimate: $m_t = \beta_1 \cdot m_{t-1} + \left(1 - \beta_1\right) \cdot g_t$ (8)

Update biased second raw moment estimate: $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (9)

Get gradients: $g_t = \nabla_\theta f_t(\theta_{t-1})$ (10)

-- α: learning rate.

-- β_1 and β_2 are the decay rates for the moment estimates.

Convolutional Neural Network – Model Compilation

- Metrics: evaluate the performance of the model during training and testing.
- Keras module: accuracy, precision, F1 score, AUC, Mean Absolute Error (MAE), etc.
- Accuracy: represents the proportion of samples that are correctly predicted to all samples.

Two-stage Training Method

- **Some factors affect the robustness of CNN**: data quantity, data quality, class distribution, noisy data, data augmentation, etc.
- **Reference literature**: Liu, Shu, and Qiang Wu. "Robust Representations in Deep Learning." DBKDA 2023 (2023): 34.
- A feature extraction part and a classification decision part.

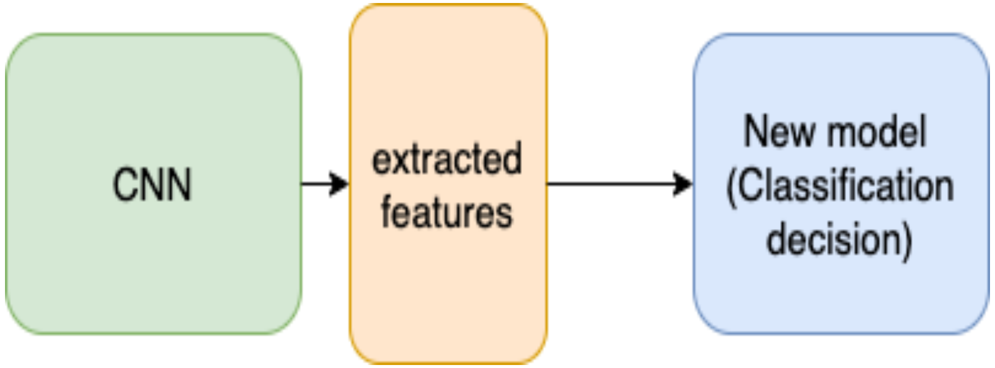


Figure 4. The workflow of the two-stage training algorithm

Two-stage Training Method

- **Stage 1: Train a CNN model**

1. Prepare the dataset *D* for image classification and preprocess the data containing labeled data $D = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the *i*th image and y_i is its class label.
2. Define a CNN architecture.
3. Train the CNN model by optimizing the parameters Θ using the Adam optimizer after certain epochs.
4. Save the trained model: M_1 .

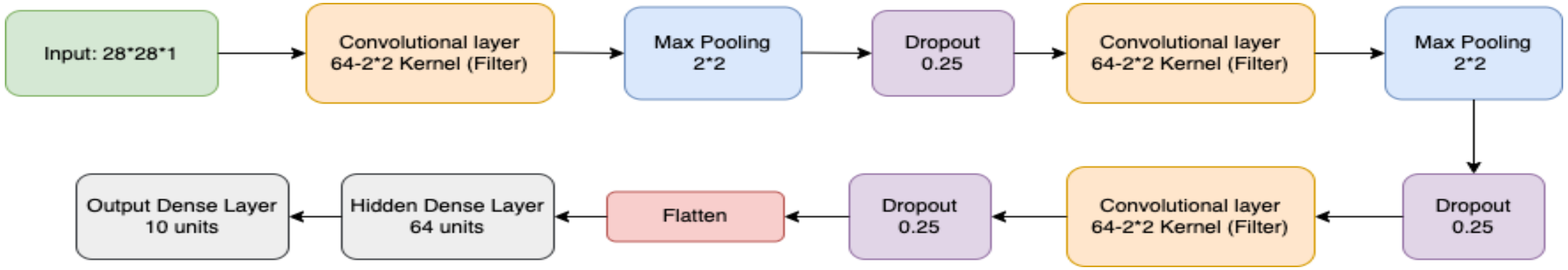
- **Stage 2: Train a new model**

1. Reload the saved model M_1 .
2. Use "Model" function to define the model M_{modify} without the last dense output layer of model M_1 , and extract features by passing the input from the model M_1 to model M_{modify}.
3. Design a single-layer neural network M_2 with the same last layer of model M_1 . The model M_2 has randomly initialized weights *W* and bias *b*, where M_2 computes the predicted class probabilities $\hat{y}_i$.
4. Train the model M_2 by updating the weights *W* and bias *b* between the inputs and last dense output layer after certain epochs.

Applications – Introduction to the four datasets

	Fashion-MNIST	Cifar-10	Dogs & Cats	Face Mask Detection
Data quantity	70,000	60,000	8,005	8,982
Number of classes	10	10	2	3
Category Balance	Yes	Yes	Yes	Yes
Input size	28×28×1	32×32×3	128×128×3	128×128×3
Ratio	1/7 (test), 6/7*20%(validation)	1/6(test), 6/7*20%(validation)	almost 20%(test), 0.8*20%(validation) (Shuffle=True)	20%(test), 0.8*20%(validation) (Shuffle=True)
Data Preprocessing	Data Normalization, label one-hot	Data Normalization, label one-hot	Data Normalization, label one-hot, OpenCV, Data Augmentation	Data Normalization, label one-hot, OpenCV, Data Augmentation

Applications - The architecture of CNN for each dataset



Input: 28*28*1

Convolutional layer 64-2*2 Kernel (Filter)

Max Pooling 2*2

Dropout 0.25

Convolutional layer 64-2*2 Kernel (Filter)

Max Pooling 2*2

Dropout 0.25

Convolutional layer 64-2*2 Kernel (Filter)

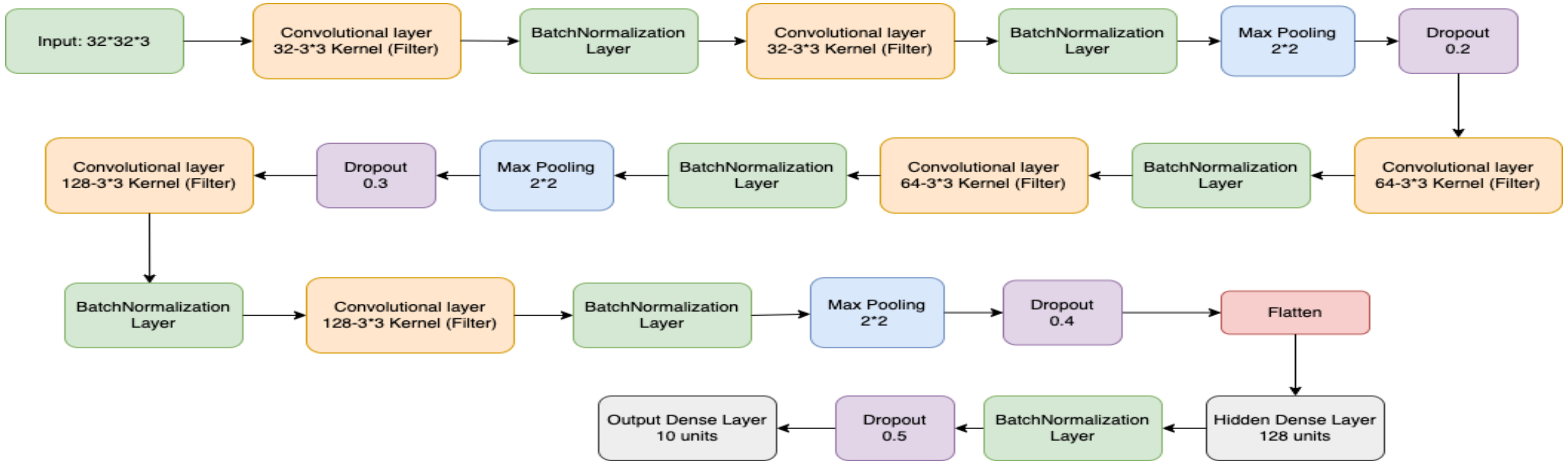
Dropout 0.25

Flatten

Hidden Dense Layer 64 units

Output Dense Layer 10 units

Figure 5. The architecture of CNN for Fashion-MNIST



Input: 32*32*3

Convolutional layer 32-3*3 Kernel (Filter)

BatchNormalization Layer

Convolutional layer 32-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2

Dropout 0.2

Convolutional layer 64-3*3 Kernel (Filter)

BatchNormalization Layer

Convolutional layer 64-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2

Dropout 0.3

Convolutional layer 128-3*3 Kernel (Filter)

BatchNormalization Layer

Convolutional layer 128-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2

Dropout 0.4

Flatten

Hidden Dense Layer 128 units

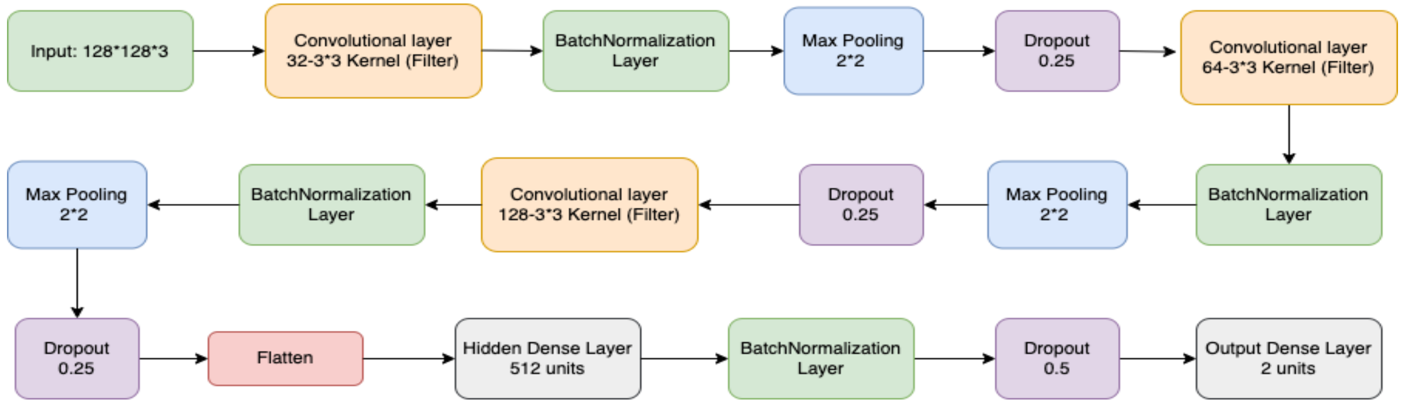
BatchNormalization Layer

Dropout 0.5

Output Dense Layer 10 units

Figure 6. The architecture of CNN for Cifar-10

Applications - The architecture of CNN for each dataset



Input: 128*128*3

Convolutional layer 32-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2

Dropout 0.25

Convolutional layer 64-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2

Dropout 0.25

Convolutional layer 128-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2

Dropout 0.25

Flatten

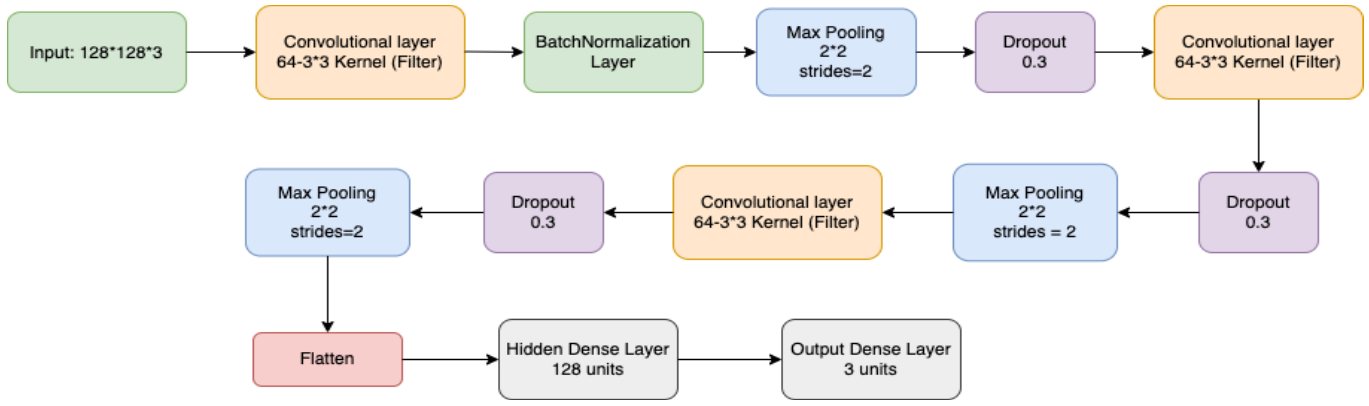
Hidden Dense Layer 512 units

BatchNormalization Layer

Dropout 0.5

Output Dense Layer 2 units

Figure 7. The architecture of CNN for Dogs & Cats



Input: 128*128*3

Convolutional layer 64-3*3 Kernel (Filter)

BatchNormalization Layer

Max Pooling 2*2 strides=2

Dropout 0.3

Convolutional layer 64-3*3 Kernel (Filter)

Dropout 0.3

Max Pooling 2*2 strides = 2

Convolutional layer 64-3*3 Kernel (Filter)

Dropout 0.3

Max Pooling 2*2 strides=2

Flatten

Hidden Dense Layer 128 units

Output Dense Layer 3 units

Figure 8. The architecture of CNN for Face Mask Detection

Applications – Some methods used in model training

- **EarlyStopping method**: check whether the metric set in the “monitor” was better than the best value found so far after each epoch finished. If there is no improvement, it determines when to stop the training process according to the parameter set in the “patience”.
- **ModelCheckPoint method**: save the best model automatically in terms of the metric set in the “monitor”. Typically, it works together with the EarlyStopping method.
- Both methods are used in four applications. The patience is 10 while the epoch is 50.
- **Learning Rate Reduction method**: allow the model to dynamically reduce the learning rate when the metric monitored stops improving, which can help the model find a better solution and avoid falling into a suboptimal solution.
- For the Face Mask Detection dataset, there are two versions designed for the same CNN architecture. The first version still uses the EarlyStopping and ModelCheckPoint methods that keep the same as the other datasets. The second version combines the EarlyStopping and Learning Rate Reduction methods to improve the performance of this model. The patience of EarlyStopping is 5, the factor of Learning Rate Reduction is 0.5 and the patience is 3.

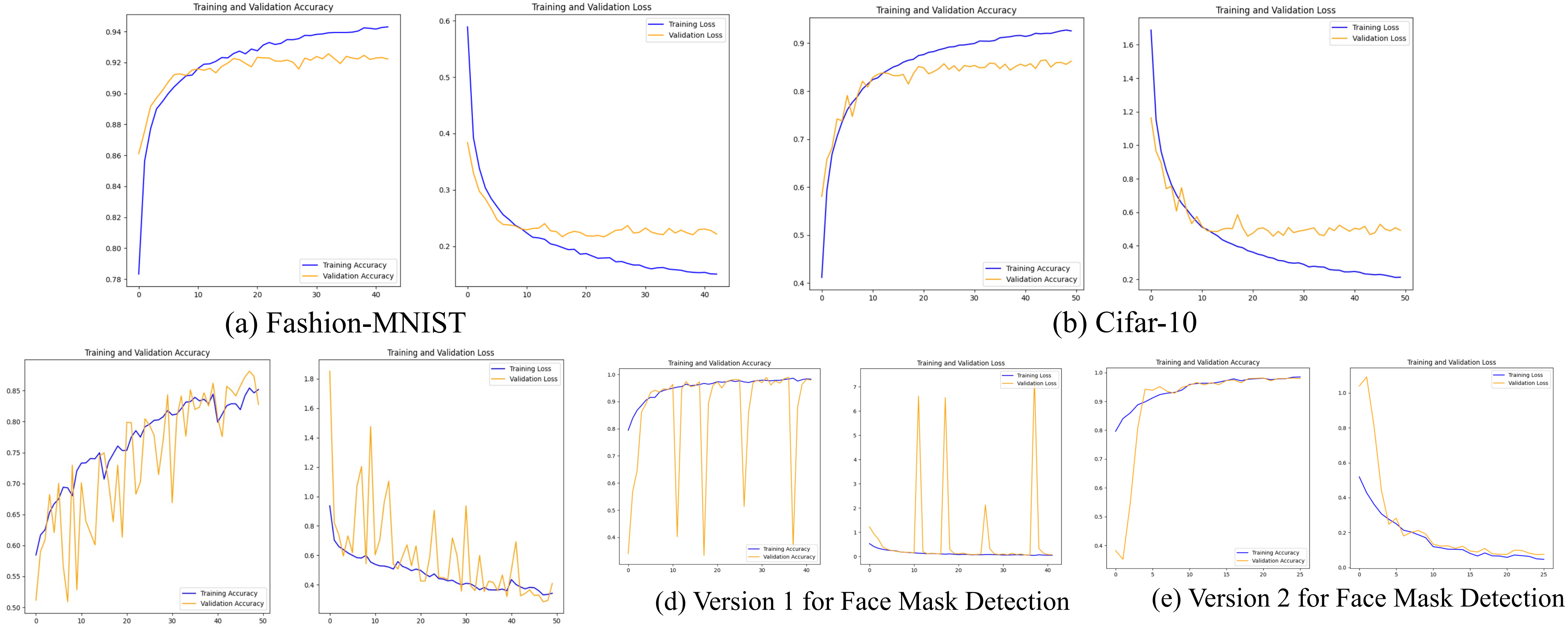
Results and Analysis

- The results were obtained by running on a GPU with the hardware name NVIDIA GeForce RTX 2080 Ti. The experiments on each dataset are repeated at least 20 times, and the results of all experiments are averaged.

Dataset Method		Fashion-MNIST	Cifar-10	Dogs & Cats	Face Mask Detection	
					Version 1	Version 2
CNN (Adam)	Training accuracy	97.03%	98.77%	82.47%	92.27%	97.28% (99.18%)
	Validation accuracy	92.10%	85.91%	80.64%	90.99%	96.11% (98.07%)
	Test accuracy	91.75%	85.66%	81.40%	90.88%	95.99% (97.87%)
CNN + second stage (Adam)	Training accuracy	97.64%	99.24%	91.62%	99.06%	99.30% (99.59%)
	Validation accuracy	92.25%	86.44%	86.75%	97.78%	98.31% (98.57%)
	Test accuracy	91.86%	86.07%	87.32%	97.59%	98.13% (98.44%)
CNN + second stage (SGD)	Training accuracy	97.49%	98.88%	91.18%	98.65%	99.03% (99.34%)
	Validation accuracy	92.30%	86.38%	86.80%	97.51%	98.00% (98.28%)
	Test accuracy	91.91%	86.06%	87.65%	97.19%	97.75% (98.10%)

Table 1. Four Datasets Testing Accuracy Results

Results and Analysis



(a) Fashion-MNIST

(b) Cifar-10

(d) Version 1 for Face Mask Detection

(e) Version 2 for Face Mask Detection

(c) Dogs & Cats

Figure 9. A random experiment of training and validation accuracy and loss of each dataset in the first stage of two-stage training

Results and Analysis

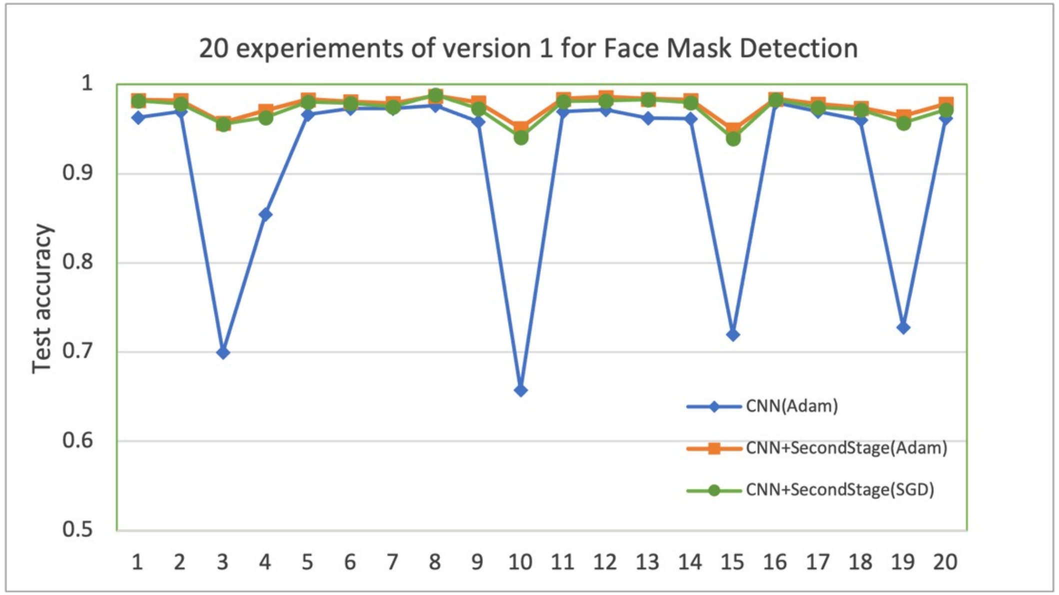


Figure 10. The test data accuracy across 20 experiments for version 1 of the Face Mask Detection dataset

Datasets	Mean of first-stage test accuracy	Mean of second-stage test accuracy	Significance Level	P(T<=t) two-tail
Fashion-MNIST	0.9175	0.9186	0.05	0.026120
Cifar-10	0.8566	0.8607	0.05	0.000320
Dogs & Cats	0.8140	0.8732	0.05	0.000016
FacemaskDetection (version1)	0.9088	0.9759	0.05	0.007079
FacemaskDetection (version2)	0.9599	0.9813	0.05	0.189317
FacemaskDetection (version2 exclude one outlier)	0.9787	0.9844	0.05	0.000004

Table 2. t-Test for Assessing the Statistical Significance of Two-Stage Algorithm Improvements

Conclusions

- The original motivation of the two-stage algorithm was to study whether two-stage training is feasible in convolutional neural networks.
- The results demonstrated that when the model training process in the one-stage algorithm is stable, the use of the two-stage algorithm has a limited ability to improve the prediction accuracy of the original model, which means that the disadvantage of this algorithm is that it cannot greatly enhance the accuracy of a well-trained model.
- But for the model with an unstable training process, the two-stage training can increase the robustness of the model.

Thank you