Project 3: Predictive Analysis

Team 2

Problem Statement

- A client wants to create an model AI program that accurately classifies images.

- Goal: build a predictive model for image classification using 50k image, 50k noisy label and 10k cleaned labels, considering balance between memory cost and running time cost.

Baseline Model

- The baseline model uses a logistic regression, which is a linear model that uses a single linear boundary to separate the classes in the feature space and treat noisy labels as clean label. The model itself is a simple and interpretable model that works well for clean linearly separable datasets.
- **Issue**: it has limited capacity to capture complex relationships between the input features and may not perform well on datasets with noise and non-linear decision boundaries.

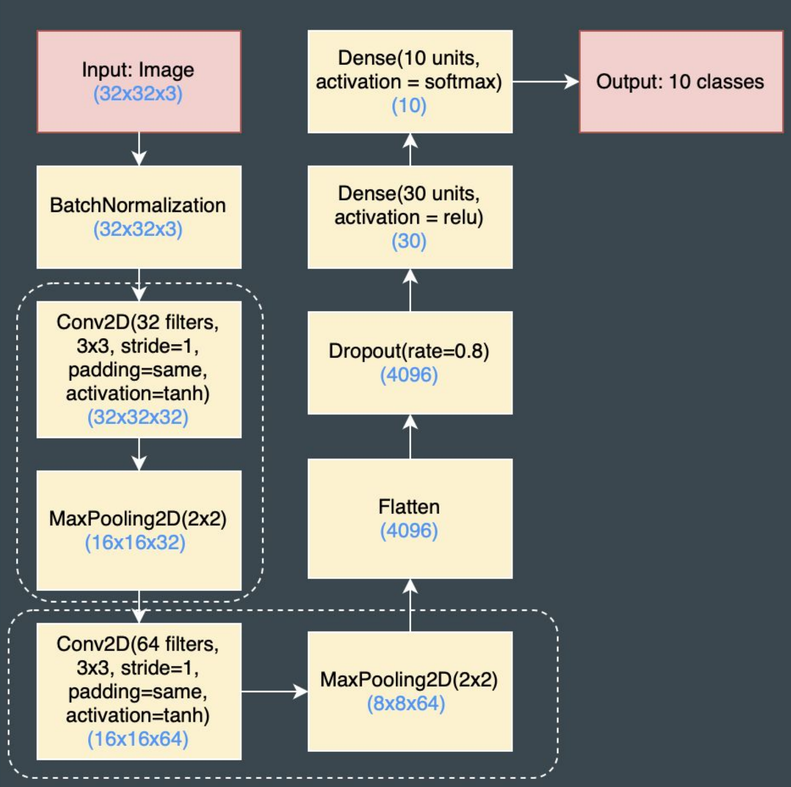
Model I - Convolution Neural Network

Data: All 50k image + 50k noisy label are used to fit the model

- Treat noisy label as cleaned

Model:

- <u>One Batch Normalization layer</u>: normalizes the input data by subtracting the mean and dividing by the standard deviation. This can help with faster convergence and better generalization.
- <u>Two CNN layers</u>:
  - <u>Conv2D</u>: apply 32(1)/64(2) 3x3 filters learned by backpropagation during training. Choose activation='tanh' because it is a popular choice for image classification tasks, and One of the main benefits of using 'tanh' is that it can produce negative outputs, which can help to better differentiate between different image features.
  - <u>MaxPooling2D</u>: reduces the spatial dimensions of the output feature maps by selecting the maximum value in each 2x2 window of the feature map. This reduces the number of parameters in the network and helps prevent overfitting.
- <u>One Flatten layer</u>: flattens the output of the previous layer to a 1D vector, which can then be fed into a fully connected layer for classification.



Input: Image (32x32x3)
BatchNormalization (32x32x3)
Conv2D(32 filters, 3x3, stride=1, padding=same, activation=tanh) (32x32x32)
MaxPooling2D(2x2) (16x16x32)
Conv2D(64 filters, 3x3, stride=1, padding=same, activation=tanh) (16x16x64)
MaxPooling2D(2x2) (8x8x64)
Flatten (4096)
Dropout(rate=0.8) (4096)
Dense(30 units, activation = relu) (30)
Dense(10 units, activation = softmax) (10)
Output: 10 classes

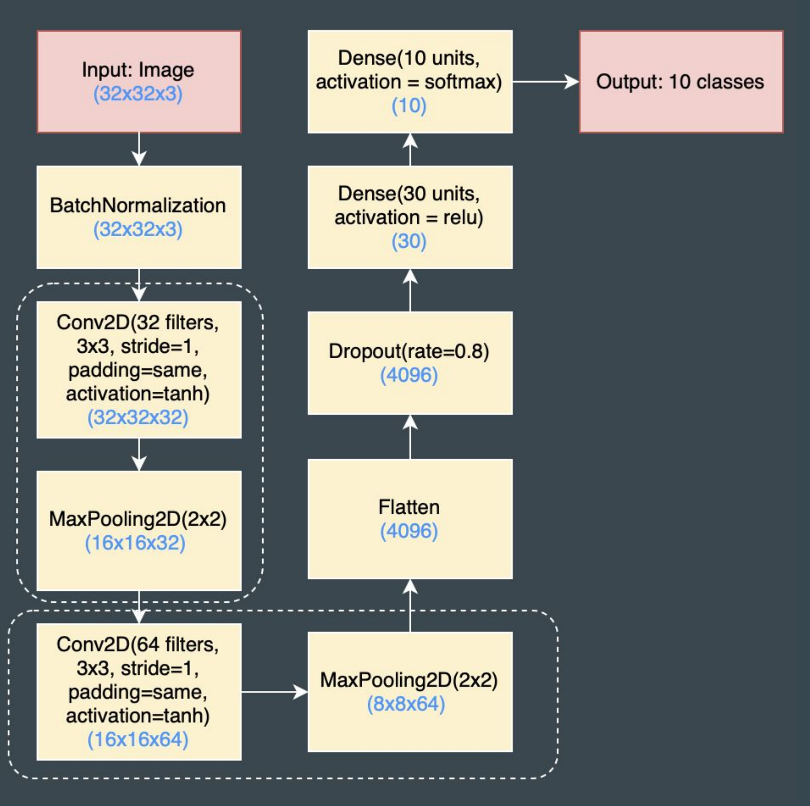
Model I - Convolution Neural Network

Model Continue:

- <u>**One Dropout layer:**</u> randomly drops out 80% of the neurons during training. This can prevent overfitting and improve generalization.
- <u>**One Dense - relu layer**</u>: takes the high-level representations learned by the convolutional layers and maps them to a lower-dimensional space. This can help to reduce the number of parameters in the model and prevent overfitting.
- <u>**One Dense - softmax layer**</u>: used to output class probabilities, as it provides a normalized way of comparing the likelihoods of different classes.

Fitting technique:

- 80% of training set and 20% of validation set
- <u>Optimizer</u>: Adam
- <u>Loss function</u>: sparse categorical cross-entropy
- <u>Early stopping callback</u>: used to prevent overfitting. It stops the training if the validation loss does not improve for a certain number of epochs.


Input: Image
(32x32x3)
BatchNormalization
(32x32x3)
Conv2D(32 filters, 3x3, stride=1, padding=same, activation=tanh)
(32x32x32)
MaxPooling2D(2x2)
(16x16x32)
Conv2D(64 filters, 3x3, stride=1, padding=same, activation=tanh)
(16x16x64)
MaxPooling2D(2x2)
(8x8x64)
Flatten
(4096)
Dropout(rate=0.8)
(4096)
Dense(30 units, activation = relu)
(30)
Dense(10 units, activation = softmax)
(10)
Output: 10 classes

Model I - Convolution Neural Network

Compare to Baseline model:

- The model I we proposed is a deep learning model that can learn more complex representations of images. Without incorporating weakly supervised learning feature into the model, we treat the noisy data as clean data when training the model. The model itself uses convolutional layers to automatically extract features from the images, and then uses fully connected layers to classify the images. This approach is generally more accurate and can handle more complex datasets.

Performance:

- Accuracy:
    - Training: 21%
    - Validation: 21%

      The accuracy slightly supersedes the basic logistic model in terms of the validation set. However, we can see that this is still not that ideal in correctly classifying the label of the image.
- Runtime: 103 seconds

Model I - Convolution Neural Network

Performance:

- Accuracy:
  - Training: 21%
  - Validation: 21%

    The accuracy slightly supersedes the basic logistic model in terms of the validation set. However, we can see that this is still not that ideal in correctly classifying the label of the image.
- Runtime: 103 seconds

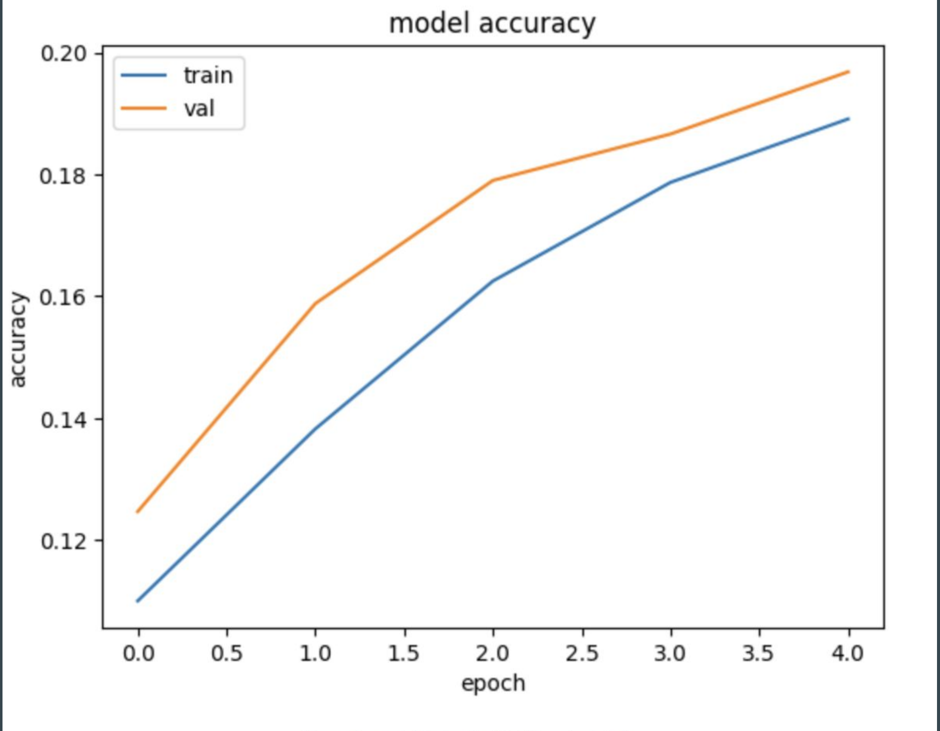
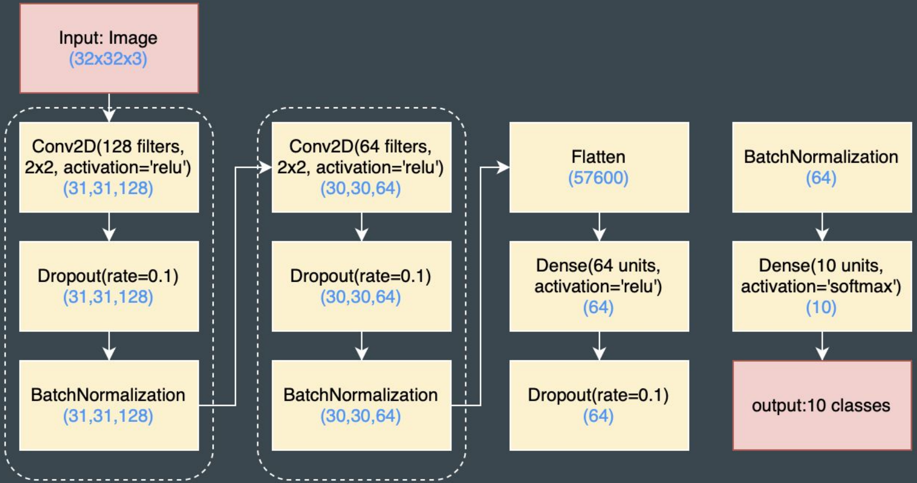


Figure 1 model trained with 50k noisy labels

Model II - CNN Improved

Model:

- Two CNN layers:
  - Conv2D: apply 128(1)/64(2) 2x2 filters learned by backpropagation during training. Choose activation='relu' because it is fast and it can introduce non-linearity between input and output. It is an improvement for model 1's CNN layer since the relationship between pixels of an image can be complex and nonlinear.
  - Dropout: randomly drops out 10% of the neurons during training. This can prevent overfitting and improve generalization.
  - Batch Normalization: helps to stabilize the distribution of activations and speed up training by reducing internal covariate shift.
- One Flatten layer
- One Dense - relu layer: By applying a fully connected layer with ReLU activation, the network can learn complex, non-linear relationships between the features extracted by the previous layers.
- One Dropout layer



- One BatchNormalization layer
- One Dense - softmax layer: produces a probability distribution over the 10 possible classes, representing the model's confidence in each class. The softmax activation function ensures that the outputs are normalized and sum to 1.

Model II - CNN Improved

Data:

- 50k images + 10k cleaned labels

Fitting steps: Weakly Supervised Learning Feature

- Train the CNN model using 10k cleaned labels with first 10k images
- Predict a clean for each of the 40k images with noisy labels
- Use the original 10k clean labels + 40k predicted labels to Train the CNN model again

Fitting technique:

- 90% of training set and 10% of validation set
- Optimizer: Nadam
- Loss Function: sparse categorical cross-entropy
- Early stopping callback: used to prevent overfitting. It stops the training if the validation loss does not improve for a certain number of epochs.

Model II - CNN Improved

Compare to Baseline model and model I:

- More accurate data: baseline model and model I only used noisy labels for training whereas model II used 10k cleaned data + 40k predicted label which is more accurate than 50k noisy labels
- More complex architecture: even though model I and model II both used CNN, model II's CNN is more complex than model I. Increasing the complexity can allow the model to learn more non-linear relationships between input and output which can lead to better performance.
- Used of regularization: model II uses various regularization techniques such as dropout and batch normalization, which can help prevent overfitting and improve the model's ability to generalize to new data.

Model II - CNN Improved

Performance:

- Accuracy:
    - Training: 88%
    - Validation: 68%
- Runtime: 1197 seconds

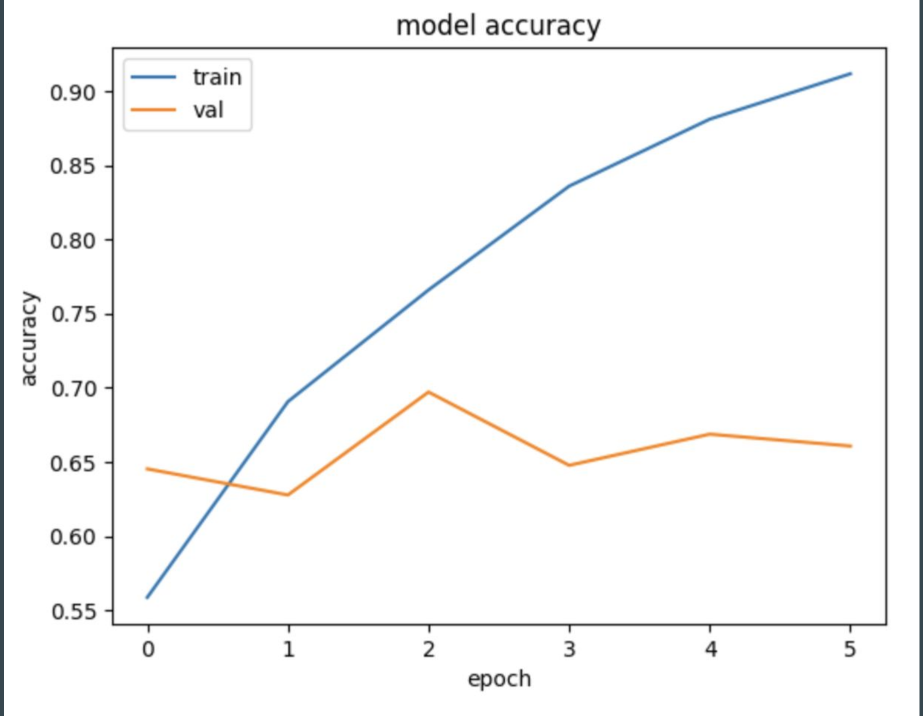


Figure 3 model trained with 10k clean labels and 40k corrected labels

Conclusion and Recommendations

Conclusion:

Overall, we find that our model II which uses a more complex CNN architecture with regularization techniques and a combination of clean and predicted labels for training, outperforms the baseline model using logistic regression and simpler CNN architecture of model I for image classification tasks. Even though the running time of model II is longer than the model I, the model II outperforms the model I significantly in terms of the accuracy as a trade-off. Considering the large volume of the data, we believe that the model II successfully achieves a balance between the running time cost and the accuracy.

Future Recommendations:

- Hyperparameter tuning: use grid-search to test different hyperparameters such as learning rate, batch size, and dropout rate can help fine-tune the model and improve its performance.
- Ensemble learning: Combining multiple models, such as model2 with other CNN architectures, can help improve the model's accuracy and robustness.