FINAL PROJECT

Report

TEAM 4

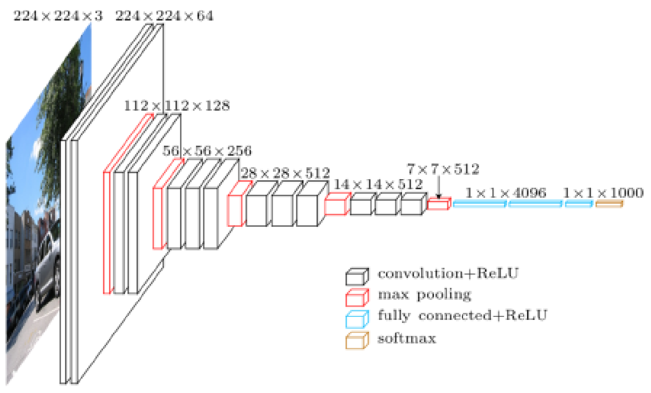
楊其龍, 黃立信, 黃政庭, 黃寶萱

-- Content --

Classifier

Methodology

In the car classifier, we build our model based on the structure of VGG-16, which is good at classification. First of all, we resize the model input size to 224*224, in this way, it will be more efficient as well effective to get the feature in the images. Our model architecture is similar to VGG-16, which includes 13 convolution layers and 3 dense layers. In those 3 dense layers, we add a dropout layer, which drop half of the neuron, between them to avoid overfitting.



Architecture of VGG16 Fig (1)

In addition, we also implement some image preprocessing. For example, we normalize each pixel in the image within 0 to 1, and reduce the dimension of the image to 224*224. In this way, we could let the model to learn and get the feature of the image easier.

How to train the model

1. Parameter Setting

   7236 training data, and 805 validation data, with batch size=32, epochs=50

2. Optimizer

   Adam with learning rate 0.00005

3. Loss Function

   We use “categorical crossentropy” as loss function

4. Evaluation

   During training session, the evaluation metrics is based on accuracy

Test Result

Average 60% accuracy

- **Demo Result**

  76.5% accuracy

Generator

- **Methodology**

  In this final project, we use Conditional Generative Adversarial Network, cGAN for short, to generate car images with specific brand or color. To handle multiple input, we use functional API to implement the discriminator, generator, and the composite model.

  For discriminator, the first input is 64x64 RGB image and the second input is an integer for the class label. The class label will encode by Embedding layer with size of 50. Concatenate two inputs into four-channel 64x64 structure and do downsampling with 3 convolutional layers. The output layer is constructed by Dense function with sigmoid function.

  For generator, 100-element point in latent space as first input and second input is class label and pass through Embedding layer. Concatenate two inputs into 129-channel 8x8 structure and upsampling with 3 convolutional layers. Then output a single 64x64 rgb image.

  For composite GAN model, combine both generator and discriminator into one large model. GAN model will take a point in latent space and class label as input and generate a prediction of whether input is real or fake.

- **How to train the model**

  1. Parameter Setting

     Discriminator(/generator) : using the LeakyReLU activation function with a slope of 0.2, using a 2×2 stride to downsample(/upsample)

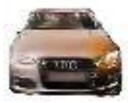
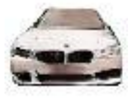
     GAN : fit for 150 epochs with batch size of 128 samples
  2. Optimizer

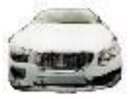
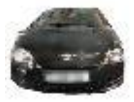
     Discriminator and generator : using Adam stochastic gradient descent with a learning rate of 0.0002 and a momentum of 0.5
  3. Loss function : all models are using ‘binary_crossentropy’ as loss function
  4. Evaluation : using ‘train_on_batch()’ to evaluate training loss with updating a single gradient on a single batch of data.

- **Bonus Part**

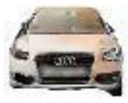
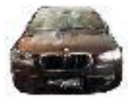
  We do the bonus part of the project by the same model. We change the number of classes of the discriminator and the generator. We choose the 14 colors which are Yellow, White, Orange, Bronze, Grey, Blue, Silver, Green, Brown, Red, Purple, Black, Beige and Unlisted. We don't use Gold color because there was too little data in gold. We also change the latent space to 140, it means that we have 14 classes and each classes has 10 results. The output is 140 64x64 images which are separated by 14 classes.

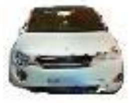
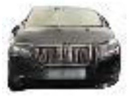
- **Test Result**

  Basic(Audi):  Basic(BMW): 

  Bonus(White):  Bonus(Black): 

- **Demo Result**

  Basic(Audi):  Basic(BMW): 

  Bonus(White):  Bonus(Black): 

Discussion

In the classifier section, we have ever got the highest classification accuracy up to 90%, it is obvious that this classifier is able to work well. In fact, what we have done is to fine tune the original VGG-16 model; moreover, because of the substantial car dataset, during our training, it really cost a lot of time as well as GPU requirement. Therefore, in the future, maybe we could design another classifier which is time-costless and GPU-costless with the high accuracy still.

In the generator section, according to our test result, it shows that our cGAN model can generate specific brand of car indeed and the performance of generating Audi images are better than BMW. In addition, we think we still can improve our model to increase the resolution of generated image so that the generated image can be clearer.

Conclusion

In this final project, we do have a deeper understanding of model architecture and usage especially how to deal with classification problems and the use of GAN model. The most difficult problems we encountered in both classifier and generator were how to improve the accuracy and deal with overfitting problem. After we repeatedly adjust the hyperparameters and complexity of the model, we finally get a model with good performance as we expected.