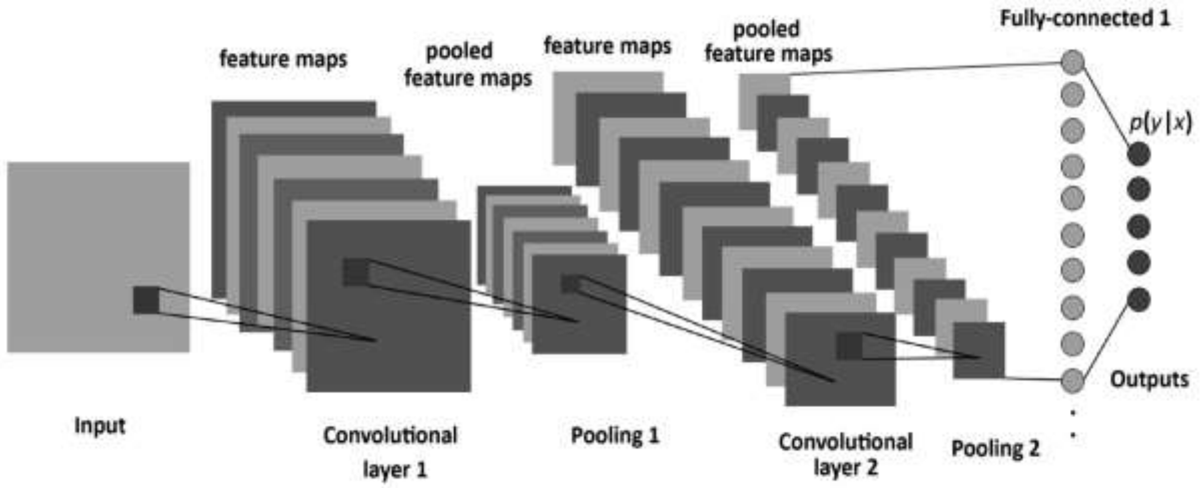
第四組 Report

組長：莊國彥

組員：何配瑜、陳遠嘉

Classifier

Methodology: CNN



Convolutional layers convolve the input and pass its result to the next layer.

Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer.

Fully connected layers connect every neuron in one layer to every neuron in another layer

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer.

Evaluation

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 680)               11141800
_________________________________________________________________
dropout_1 (Dropout)          (None, 680)               0
_________________________________________________________________
dense_1 (Dense)              (None, 30)                20430
=================================================================
Total params: 11,181,046
Trainable params: 11,181,046
Non-trainable params: 0
```

In our classifier, we divided the model into 2 layers, with Dropout(0.2) for each. And train it with 22 epoch with 90000 samples.

Test Result

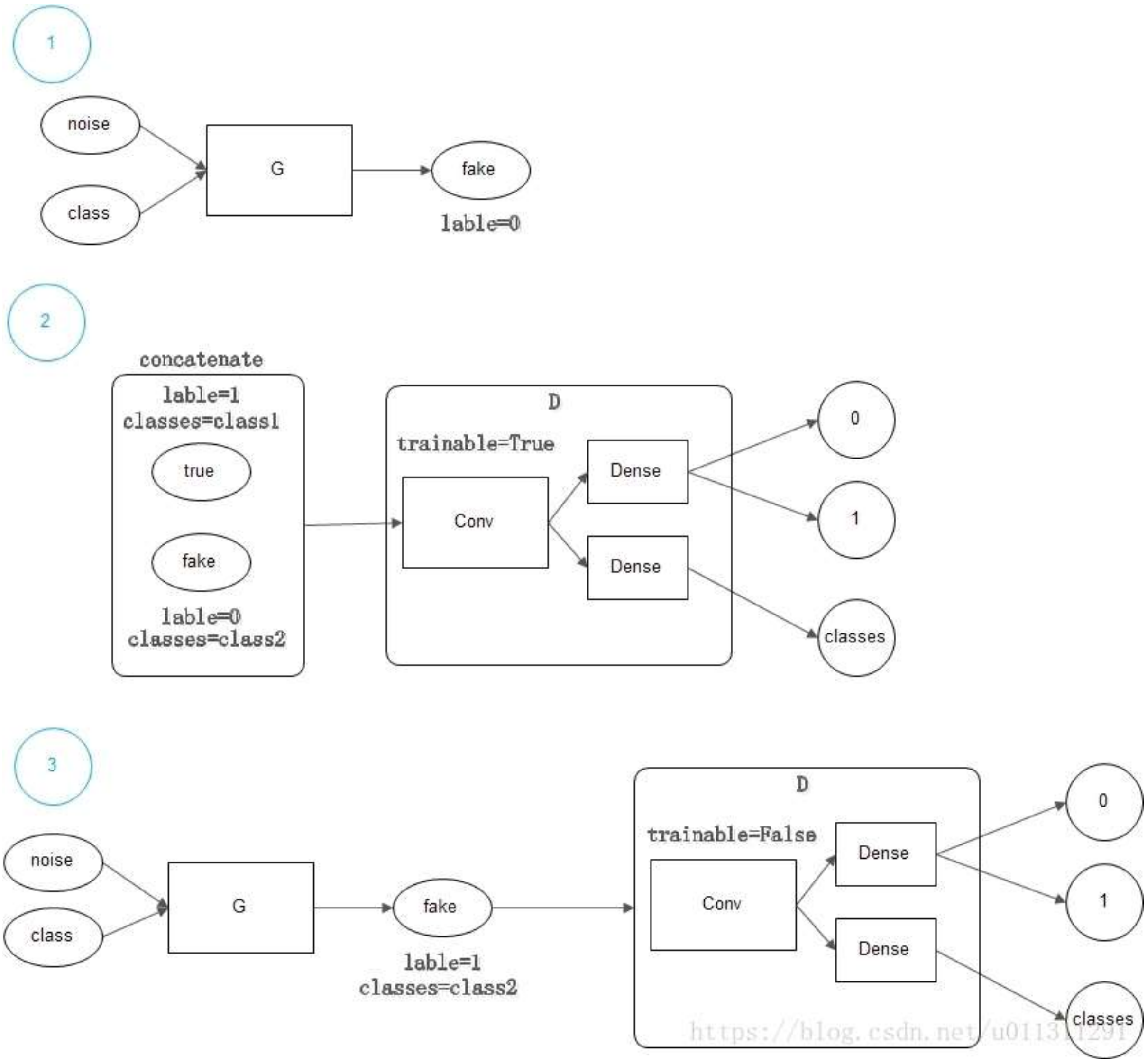
- Accuracy: 0.9951

Demo Result

- Classifier: 34/40

Generator

Methodology-ACGAN



In generator, we use acgan as our methodology. It is construct with cGan and sGan. By using ACGAN, we hope that our discriminator can classify the real data and fake data better while the generator can generate better.

Evaluation

```
model = Sequential()

model.add(Dense(128 * 7 * 7, activation="relu", input_dim=self.latent_dim))
model.add(Reshape((7, 7, 128)))
model.add(BatchNormalization(momentum=0.8))
model.add(UpSampling2D())
model.add(Conv2D(128, kernel_size=3, padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(momentum=0.8))
model.add(UpSampling2D())
model.add(Conv2D(64, kernel_size=3, padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(momentum=0.8))
model.add(Conv2D(self.channels, kernel_size=3, padding='same'))
model.add(Activation("tanh"))

model.summary()

noise = Input(shape=(self.latent_dim,))
label = Input(shape=(1,), dtype='int32')
label_embedding = Flatten()(Embedding(self.num_classes, 100)(label))

model_input = multiply([noise, label_embedding])
img = model(model_input)
```

```
def build_discriminator(self):

    model = Sequential()

    model.add(Conv2D(16, kernel_size=3, strides=2, input_shape=self.img_shape, paddir
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(Conv2D(32, kernel_size=3, strides=2, padding="same"))
    model.add(ZeroPadding2D(padding=((0,1),(0,1))))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Conv2D(64, kernel_size=3, strides=2, padding="same"))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Conv2D(128, kernel_size=3, strides=1, padding="same"))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.summary()

    img = Input(shape=self.img_shape)
```

Test Result

- Accuracy : 0.9412

Demo Result

- Generator : 0/100

Discussion

- **1.** Does GAN have a requirement for the distribution of noise z? Common distribution?

  Generally there is no special requirement, usually Gaussian distribution, uniform distribution. The dimension of the noise must reach at least the intrinsic dimension of the data manifold,
  To generate enough diversity; mnist is about 6 dimensions, CelebA is about 20 dimensions

- **2.** How many updates of G and D in one iteration? In order to make G learn better, can we make G more updated?

  D is updated more times, if G is updated too many times it will cause more shortages.

- **3.** What is the effect of adding a batch normalization layer to the GAN?

  more stable:
  Solve the random initialization parameter is not ideal,
  To prevent gradient explosions, but only to reduce the probability, other uncontrollable factors may still cause gradient explosions.

GAN Conclusion

In this paper we introduce new methods for the improved training of generative adversarial networks for image synthesis. We construct a variant of GANs employing label conditioning that results in 128×128 resolution image samples exhibiting global coherence. We expand on previous work for image quality assessment to provide two new analyses for assessing the discriminability and diversity of samples from class-conditional image synthesis models. These analyses demonstrate that high resolution samples provide class information not present in low resolution samples. Across 1000 ImageNet classes, 128×128 samples are more than twice as discriminable as artificially resized 32×32 samples. In addition, 80% of the classes have samples exhibiting diversity comparable to real ImageNet data.

In this project, we use GAN to generate pictures and throw the trained generator into it. This modification to the standard GAN formulation produces excellent results and appears to stabilize training. Along with our proposed methods for measuring the extent to which a model makes use of its given output resolution, methods for measuring perceptual variability of samples from the model, and a thorough experimental analysis of a generative model of images that creates 128×128 samples from all 1000 ImageNet classes.

We train several models on the ImageNet data set . Broadly speaking, the architecture of the generator G is a series of “deconvolution” layers that transform the noise z and class c into an image. We performed the same analysis, trained to 64×64 spatial resolution. This model achieved less discriminability than a 128×128. Accuracies from the 64×64 model plateau at a 64×64 spatial resolution consistent with previous results. Finally, the 64×64 resolution model achieves less discriminability at 64 spatial resolution than the 128×128 model.