1. Introduction (5%)

   透過對抗式網路來學出 ground turth data 的分佈，discriminator 負責判斷輸入的圖片有多少機率會是真的，generator 希望能夠騙過 discriminator 讓他以為 generator 產生的圖片是真的。

2. Implementation details (15%)

   - Describe how you implement your model, including your choice of cGAN, model architectures, and loss functions. (10%)

     Architecture of conditional GAN 使用 Conditional GAN。

     > 對於 generator 就將 condition concat 在那一條 noise vector 後面，對於 discriminator 就把c ondition 轉成 (batchsize, 1, height, width) 的大小並 concat 在圖片上當作第四個 channel（原先三個 channel 為 RGB ）。[Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.]

     Design of generator and discriminators 使用 DCGAN。

     > 因為要處理的資料是圖片，將透過在 generator 裡放 convTrans 與在 discriminator 裡放 conv layer，分別從 noise vector 層層轉出圖片、從圖片層層轉至一個機率。

     Loss function 依序嘗試了 GAN、WGAN、WGANGP，原理闡述如下。

     > 先備知識
     >
     > > Binary cross entropy 的公式：
     > >
     > > $$\text{binary cross entropy } H(p,q) = -E_{x\sim p(x)} \log q(x) = -\sum_{x=1}^{2}[p(x)\log q(x)] = [p(x)\log q(x) + (1-p(x))\log(1-q(x))]$$
     > >
     > > Kullback-Leibler divergence 的缺點一是不對稱，缺點二是當同時有其中一個分佈的部分為 0，則 KLD 有可能會變得無窮大：
     > >
     > > $$KL(p(x)||q(x)) = \int p(x)\log\frac{p(x)}{q(x)}dx$$
     > > $$\begin{cases} \text{when } p(x)\to 0, q(x)\to 1, 0*\log\frac{0}{1}\to 0 \left(\because \lim_{x\to 0}(x\log x) = \lim_{x\to 0}\frac{\log x}{x^{-1}} = \lim_{x\to 0}\frac{x^{-1}}{-x^{-2}} = \lim_{x\to 0} -x = 0\right) \\ \text{when } p(x)\to 1, q(x)\to 0, 1*\log\frac{1}{0}\to +\infty \end{cases}$$
     > >
     > > Jensen–Shannon divergence 的缺點是當兩個分佈完全不重疊時，無論兩分佈距離有多遠，JSD 衡量出的距離永遠都是 $\log 2$ ：
     > >
     > > $$\begin{aligned} JS(p(x)||q(x)) &= \frac{1}{2}KL((p(x)||m(x))) + \frac{1}{2}KL(q(x)||m(x)),\ \text{where } m(x) = \frac{q(x)+p(x)}{2} \\ &= \frac{1}{2}\sum_x p(x)\log\frac{p(x)}{(q(x)+p(x))/2} + \frac{1}{2}\sum_x q(x)\log\frac{q(x)}{(q(x)+p(x))/2} \\ &= \frac{1}{2}\sum_x p(x)\left(\log\frac{p(x)}{q(x)+p(x)} + \log 2\right) + \frac{1}{2}\sum_x q(x)\left(\log\frac{q(x)}{q(x)+p(x)} + \log 2\right) \\ &= \frac{1}{2}\sum_x p(x)\log\frac{p(x)}{q(x)+p(x)} + \frac{1}{2}\log 2\sum_x p(x) + \frac{1}{2}\sum_x q(x)\log\frac{q(x)}{q(x)+p(x)} + \frac{1}{2}\log 2\sum_x q(x), \\ &\quad \text{since } \sum_x p(x) = \sum_x q(x) = 1 \\ &= \frac{1}{2}\sum_x p(x)\log\frac{p(x)}{q(x)+p(x)} + \frac{1}{2}\sum_x q(x)\log\frac{q(x)}{q(x)+p(x)} + \log 2 \\ &\quad \text{當 } p(x) \text{ 與 } q(x) \text{ 不重疊時, 該不重疊的部分將會有 } p(x) \text{ 或 } q(x) \text{ 其中一方趨近於 } 0 \\ \Rightarrow &\begin{cases} \text{when } p(x)\to 0, q(x)\neq 0, \frac{1}{2}\sum_x 0*\log\frac{0}{q(x)+0} + \frac{1}{2}\sum_x q(x)\log\frac{q(x)}{q(x)+0} \to 0 \\ \text{when } p(x)\neq 0, q(x)\to 0, \frac{1}{2}\sum_x p(x)\log\frac{p(x)}{0+p(x)} + \frac{1}{2}\sum_x 0\log\frac{0}{0+p(x)} \to 0 \end{cases} \\ \Rightarrow &\quad \text{此時 } JS(p(x)||q(x)) = 0 + \log 2 = \log 2 \end{aligned}$$
     > >
     > > Wasserstain distance （Earth mover distance）的優點是即使兩個分佈沒有重疊的部分，仍然可以合理的算出兩分佈之間的距離
     > >
     > > $$\begin{aligned} W(p,q) &= \inf_{\gamma\in\prod(p,q)} E_{(x,y)\sim\gamma}[||(x-y)||] \\ &= \inf_{\gamma\in\prod(p,q)} \sum_{(x,y)} \gamma(x,y)||(x-y)|| \end{aligned}$$

> **Example 1** (Learning parallel lines). Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,
>
> - $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
> - $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
> - $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
>
> 【GANs学习笔记】（六）JS Divergence不是最好的Divergence
>
> GAN：两者分布不重合JS散度为log2的数学证明

GAN 介紹

> 原始 GAN 的 loss 為以下定義，z 為 noise，通常產自 gaussian distribution 或 uniform distribution，$D(x;\theta^{(D)}))$ 為 discriminator (binary classifier)，負責判斷輸入的資料是真的的機率有多少，$G(z;\theta^{(G)})$ 為 generator，負責產生出真假難辨的資料。
>
> $$\begin{aligned}
> \text{Discriminator cost} &= -E_{x\sim p_{\text{data}}} \log D(x;\theta^{(D)}) - E_{z\sim p_z} \log\Big(1 - D(G(z;\theta^{(G)}),\theta^{(D)})\Big) \\
> &= H(p_{\text{data}}(x), D(x;\theta^{(D)})) + H(p_z(z), 1 - D(G(z;\theta^{(G)}),\theta^{(D)})), \\
> &\quad \text{since the probability of } x \text{ generating from } p_{\text{data}} \text{ is 1,} \\
> &\quad \text{and the probability of } z \text{ generating from } p_z \text{ is 1.} \\
> &= H(1, D(x;\theta^{(D)})) + H(1, 1 - D(G(z;\theta^{(G)}),\theta^{(D)})) \\
> &= H(1, D(x;\theta^{(D)})) + H(0, D(G(z;\theta^{(G)}),\theta^{(D)})) \\
> \text{Generator cost} &= -\text{Discriminator cost} = E_{x\sim p_{\text{data}}} \log D(x;\theta^{(D)}) + E_{z\sim p_z} \log\Big(1 - D(G(z;\theta^{(G)}),\theta^{(D)})\Big), \\
> &\quad \text{since } E_{x\sim p_{\text{data}}} \log D(x;\theta^{(D)}) \text{ is only relevant to the ground truth distribution and discriminat} \\
> &\quad \text{it's irrelevant to generator.} \\
> &= E_{z\sim p_z} \log\Big(1 - D(G(z;\theta^{(G)}),\theta^{(D)})\Big) \\
> &= -H(1, 1 - D(G(z;\theta^{(G)}),\theta^{(D)})) = -H(0, D(G(z;\theta^{(G)}),\theta^{(D)}))
> \end{aligned}$$
>
> 目標是希望可以訓練出一個 optimal generator 可以產生出非常接近 ground truth 的資料，使得 discriminator 覺得該資料真假難辨
>
> $$\theta^{(G)*} = \arg\min_{\theta^{(G)}} \max_{\theta^{(D)}} E_{x\sim p_{\text{data}}} \log D(x;\theta^{(D)}) + E_{z\sim p_z} \log\Big(1 - D(G(z;\theta^{(G)}))\Big)$$
>
> ```
> lossD = criterion(predict_real, real_noise) + criterion(predict_fake, fake_noise) # H(1, D(x)) +
> H(0, D(G(z))
> lossG = criterion(predict_chaotic_real, real) # H(1, D(G(z))
> ```

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

[UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS]

[DCGAN pytorch tutorial]

[How to use BinaryCrossEntropy (intutively) for Generator Network in DCGAN model?]

可是這樣的 loss 會有問題，即要達到訓練最佳效果必須使 discriminator 訓練成不接近最佳也不接近最差的 discriminator。當 discriminator 離 optimal discriminator 很遠時，generator 根據 discriminator 來更新就會產出非常不穩定的結果；當訓練到 optimal 的 discriminator 時，generator cost 基本上會變成定值，因為在高維空間中 p_g 對應的 manifold 與 p_{data} 對應的 manifold 重疊的機率基本上是 0 （想像在三維空間（high dimensional space） 裡面隨機取出兩個二維的平面（manifold），而該兩平面重疊在一起的機率，或是在二維空間裡面隨機取出兩個一維的點，而該兩點重疊在一起的機率），而就算相交，交界部分的在高維空間的測度（measure）也會是 0 （想像要對一個 probability density function 描繪的分佈中的某個"點"積分的結果，或是在三維空間裡二維的平面的體積、在二維空間中一維的點的面積），再想想 JSD 的缺點，如果 p_g 與 p_{data} 完全不重疊的話，JSD 永遠會是 $\log 2$，如此一來 generator cost 就會是 constant，因此對定值的 generator cost 微分就會得到 0 （梯度消失）而無法更新 generator cost，

$$\begin{aligned}
\text{Discriminator cost} &= -E_{x\sim p_{\text{data}}} \log D(x;\theta^{(D)}) - E_{z\sim p_z} \log\left(1 - D(G(z;\theta^{(G)});\theta^{(D)})\right) \\
&= -E_{x\sim p_{\text{data}}} \log D(x;\theta^{(D)}) - E_{x\sim p_g} \log\left(1 - D(x;\theta^{(D)})\right)
\end{aligned}$$

if we take the gradient of discriminator cost w.r.t. $D(x;\theta^{(D)})$, we will get the optimal discriminator $D_G^*(x)$

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

then the optimal generator's parameter $\theta^{(G)*}$ will be

$$\begin{aligned}
\theta^{(G)*} &= \arg\min_{p_g} E_{x\sim p_{\text{data}}} \log D_G^*(x) + E_{x\sim p_g} \log\left(1 - D_G^*(x)\right) \\
&= \arg\min_{p_g} E_{x\sim p_{\text{data}}} \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} + E_{x\sim p_g} \log\left(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}\right) \\
&= \arg\min_{p_g} E_{x\sim p_{\text{data}}} \log\left(\frac{p_{\text{data}}(x)}{[p_{\text{data}}(x) + p_g(x)]/2} * \frac{1}{2}\right) + E_{x\sim p_g} \log\left(\frac{p_g(x)}{[p_{\text{data}}(x) + p_g(x)]/2} * \frac{1}{2}\right) \\
&= \arg\min_{p_g} -\log(4) + \text{KL}\left(p_{\text{data}} || \frac{p_{\text{data}} + p_g}{2}\right) + \text{KL}\left(p_g || \frac{p_{\text{data}} + p_g}{2}\right) \\
&= \arg\min_{p_g} -\log(4) + 2 * \text{JSD}(p_{\text{data}} || p_g)
\end{aligned}$$

notice that $E_{x\sim p_g} \log\left(1 - D_G^*(x)\right) = -\log(4) + 2 * \text{JSD}(p_{\text{data}} || p_g) - E_{x\sim p_{\text{data}}} \log D_G^*(x) \leftarrow (1.)$

就算將 GAN 的 generator cost 改良成 $E_{z\sim p_z}[-\log D(G(z;\theta^{(G)}),\theta^{(D)})] = E_{x\sim p_g}[-\log D(x;\theta^{(D)})]$ 還是會有問題

$$\begin{aligned}
\text{Generator cost} &= E_{z\sim p_z}[-\log D(G(z;\theta^{(G)}),\theta^{(D)})] \\
&= -E_{z\sim p_z}[\log D(G(z;\theta^{(G)}),\theta^{(D)})] \\
&= H(1, D(G(z;\theta^{(G)}),\theta^{(D)}))
\end{aligned}$$

在得到 optimal discriminator 的情況下，optimal generator 會由 KLD 與 JSD 共同計算而來，想想 KLD 的缺點，當 $p_g(x) \to 0, p_{\text{data}}(x) \to 1$ 時，KLD $\to 0$，而當 $p_g(x) \to 1, p_{\text{data}}(x) \to 0$ 時，KLD $\to \infty$，明顯後者得到的懲罰比前者大，因此 generator 就會轉而專注於避免產生離 ground truth 很遠的資料，改一直生成相對安全卻看起來很像的資料，造成 collapse mode 的問題。

$$
\begin{aligned}
\theta^{(G)*} &= \arg\min_{p_g} E_{x\sim p_g}[-\log D^*(x)] \\
&= E_{x\sim p_g}\left[-\log D^*(x) + \log\left(1 - D^*(x)\right) - \log\left(1 - D^*(x)\right)\right] \\
&= E_{x\sim p_g}\left[\log \frac{1 - D^*(x)}{D^*(x)} - \log\left(1 - D^*(x)\right)\right] \\
&= E_{x\sim p_g} \log \frac{1 - D^*(x)}{D^*(x)} - E_{x\sim p_g} \log\left(1 - D^*(x)\right) \\
&= E_{x\sim p_g} \log \frac{p_g(x)/(p_{\text{data}}(x) + p_g(x))}{p_{\text{data}}(x)/(p_{\text{data}}(x) + p_g(x))} - E_{x\sim p_g} \log\left(1 - D^*(x)\right) \\
&= E_{x\sim p_g} \log \frac{p_g(x)}{p_{\text{data}}(x)} - E_{x\sim p_g} \log\left(1 - D^*(x)\right) \\
&= KL(p_g||p_{\text{data}}) - E_{x\sim p_g} \log\left(1 - D^*(x)\right) \because (1.) \\
&= KL(p_g||p_{\text{data}}) + \log(4) - 2 * \text{JSD}(p_{\text{data}}||p_g) + E_{x\sim p_{\text{data}}} \log D_G^*(x), \\
&\quad \text{since } E_{x\sim p_{\text{data}}} \log D_G^*(x) \text{ is irrelevant to generator} \\
&= KL(p_g||p_{\text{data}}) - 2 * \text{JSD}(p_{\text{data}}||p_g)
\end{aligned}
$$

[Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.](#)

[機器學習] GAN 筆記

WGAN 介紹

因此就衍生了 WGAN 使用 Wasserstain distance 來計算 discriminator cost 與 generator cost，這裡省略計算過程，因為上課投影片 L20-DeepGenerativeModels 有寫，總之目的就是要透過 minimize earth mover distance 來減少 generator 的分佈與 ground truth data 的分佈間的差距，而 earth mover distance 則是會找出從一個分佈搬家到另一個分佈所花的最小 effort，這可以透過 dual problem 求某個算式的最大值得出，整體算式如下。WGAN 與原始 GAN 的差別就在於 1) discriminator 從 binary classifier 變成 critic，因此最後 discriminator 最後一層的 activation function 也從 sigmoid 變成 linear(通常會直接把 sigmoid 拿掉)，2) discriminator 與 generator 的 cost 不取 log，3) 需要對 discriminator parameter ω 做 clip 使符合 1-Lipschitz 的條件，4) optimizer 推薦使用 RMSProp 或 SGD，而不建議使用 Adam 或 momentum。

$$
\begin{aligned}
\theta^* = \arg\min_{\theta} \max_{\omega\in W} & E_{x\sim p_{\text{data}}} f_\omega(x) - E_{z\sim p(z)} f_\omega(g_\theta(z)), \\
& \text{where } f_\omega \text{ and } g_\theta \text{ represents the discriminator(critic) and generator respectively,} \\
& \text{and } f_\omega \text{ is a family of 1-Lipschitz functions (any two points' absolute slope of function } f_\omega \le 1), \\
& \frac{||f_\omega(x_i) - f_\omega(x_j)||}{||x_i - x_j||} \le 1
\end{aligned}
$$

```
lossD = -predict_real.mean() + predict_fake.mean()
# ... clip
for param in model.discriminator.parameters():
  param.data.clamp_(-args.clipRatio, args.clipRatio)
lossG = -predict_chaotic_real.mean()
```

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

[Wasserstein GAN]

[令人拍案叫绝的Wasserstein GAN]

WGANGP

但透過 clip 來做 1-Lipschitz 效果並不是很好，所以改用懲罰的方式來達到這個目標。

```
eps = torch.rand(batch_size, 1, 1, 1).to(device)
interpolot_image_batch = (eps * real_image_batch.data + (1-
eps)*fake_image_batch.data).requires_grad_(True) #x
predict_interplot = model.discriminator.forward(interpolot_image_batch, condition_batch) #y
gradient = torch.autograd.grad(outputs=predict_interplot, inputs=interpolot_image_batch,
                               grad_outputs=torch.ones(predict_interplot.size()).to(device),
                               create_graph=True, retain_graph=True)[0] #dydx
norm_gradient = torch.sqrt(torch.sum(gradient.view(batch_size, -1) ** 2, dim=1))
gradient_penalty = torch.mean((norm_gradient - 1) ** 2)
lossD = -predict_real.mean() + predict_fake.mean() + args.gradientPenaltyWeight*gradient_penalty
lossG = -predict_chaotic_real.mean()
```

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

[Improved Training of Wasserstein GANs]

- Specify the hyperparameters (learning rate, epochs, etc.) (5%)

  epochsize 上限設 300，但常常跑到一半發現已經收斂就提前砍掉了；

  batchsize 設為 64；

  image_side_length （圖片邊長）設為 64；

  noise_size（noise vector 的維度）設為 100；

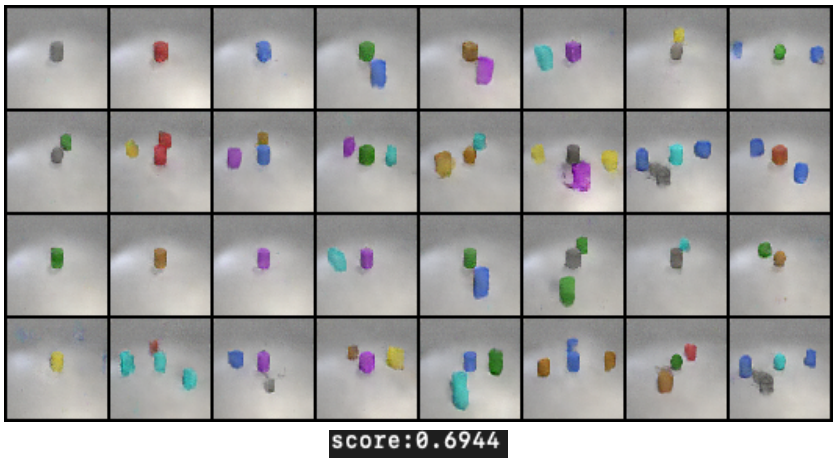
  condition_embedding_size（決定 condition 要跟輸給 generator 的 noise concat 在一起時，condition 要轉成多少長度）設為100；

  generator_RepTimesBalance、discriminator_RepTimesBalance 用來決定 generator、discriminator 要 backward 幾次，對於 GAN 分別設 3、1，這是為了避免 discriminator 初期太強，導致 generator 無法有效學習，對於 WGAN 與 WGANGP 分別設5、1，這是為了產生良好的 critic 以盡可能準確衡量 generator 產生的圖片與 ground truth 圖片的差距。

  gradientPenaltyWeight（也就是 WGANGP 裡用來懲罰違反 Lipschitz 的權重）設為 10；

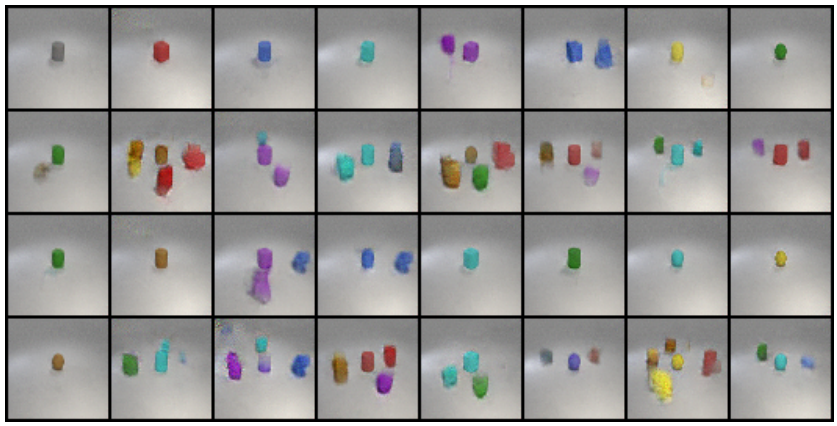
  learning rate 與 optimizer 在不同架構下設置也不同， DCGAN+GAN / DCGAN+WGAN / DCGAN+WGANGP 分別會使用 Adam / RMSProp / Adam optimizer，而 learning rate 分別是 2e-4 / 5e-5 / 1e-4，另外 Adam 還會使用另外兩個參數 (beta1, beta2)=(0.5, 0.9)。

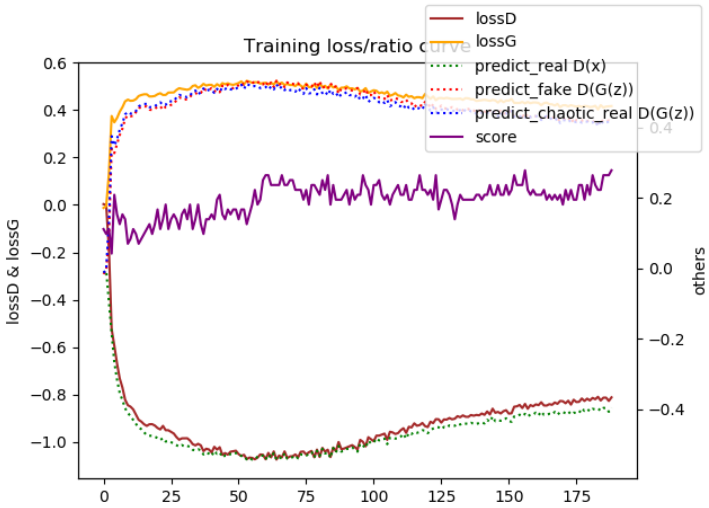
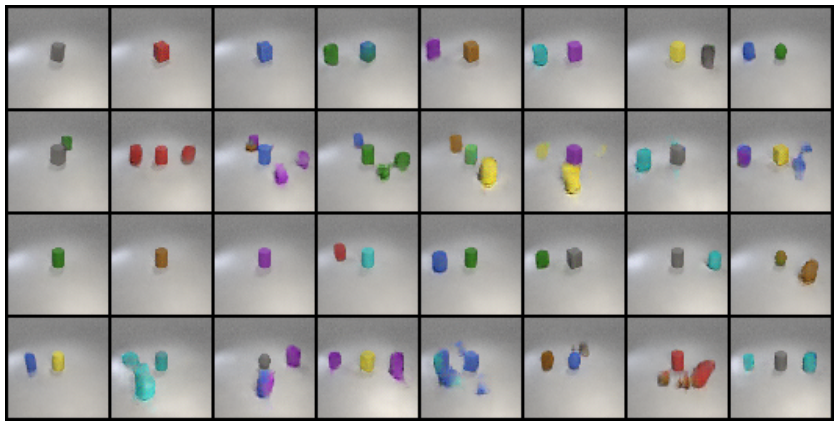
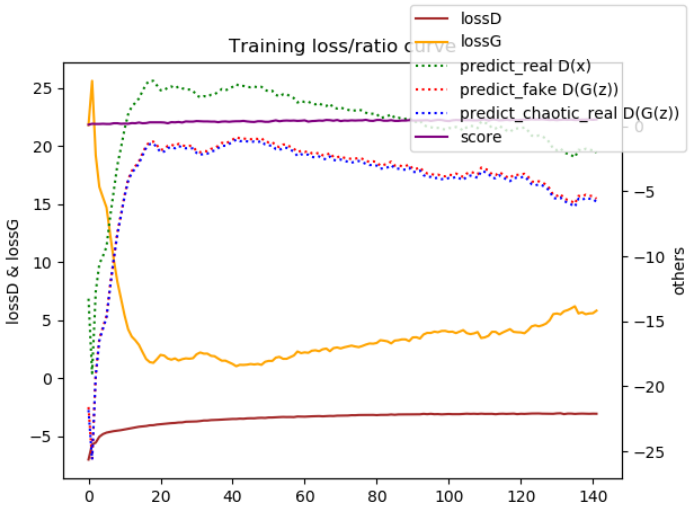
3. Results and discussion (20%)
   - Show your results based on the testing data. (5%) (including images)



     準確率為 0.69。

   - Discuss the results of different models architectures. (15%) For example, what is the effect with or without some specific loss terms, or what kinds of condition design is more effective to help cGAN.
     1. 訓練初期出現 discriminator 太強大導致 generator 學不了的情況，解法如下
        - 使 generator 的訓練次數比 discriminator 多，分別為 1、3~5
        - 使 generator 的學習率比 discriminator 低，分別為 1e-4、4e-4
        - 將 label (fake or real) 加上 noise 餵給 discriminator (並非餵進 real image 混上一點 fake image)
        - 把 loss 從最原始的 GAN 改成 WGAN
     2. 純 GAN 很難收斂，但是一旦收斂了，準確率本身就不錯看
     3. WGAN 比 GAN 好收斂，訓練過程中 lossD 與 lossG 不管是正的還是負的，最終都會往零靠近，但準確率不高。





4. WGANGP 比 GAN 好收斂，準確率比 WGAN 高，比純 GAN 低。