



به نام خدا  
دانشگاه تهران  
دانشکده مهندسی برق و کامپیوتر



## درس شبکه‌های عصبی و یادگیری عمیق

### تمرین ششم

نام و نام خانوادگی	سارا رستمی – امین شاهچراغی
شماره دانشجویی	۸۱۰۱۹۹۱۹۶ – ۸۱۰۱۰۰۳۵۵
تاریخ ارسال گزارش	۱۴۰۱.۱۱.۰۲

## فهرست

- پاسخ ۱. شبکه‌های مولد تخصصی کانولوشنال عمیق ..... ۵
- ۱-۱. پیاده‌سازی مولد تصویر با استفاده از شبکه‌های مولد تخصصی کانولوشنال عمیق ..... ۵
- ۲-۱. ارزیابی شبکه ..... ۹
- ۳-۱. پایدارسازی شبکه ..... ۹
- پاسخ ۲. شبکه متخاصم مولد طبقه‌بند کمکی و شبکه Wasserstein ..... ۱۲
- ۱-۲. شبکه متخاصم مولد طبقه‌بند کمکی ..... ۱۲
- ۱-۱-۲. پیاده‌سازی شبکه ..... ۱۲
- ۲-۱-۲. ارزیابی شبکه ..... ۱۵
- ۳-۱-۲. پایدارسازی شبکه ..... ۱۶
- ۲-۲. شبکه متخاصم مولد Wasserstein ..... ۱۹
- ۱-۲-۲. پیاده‌سازی شبکه ..... ۲۰
- ۲-۲-۲. ارزیابی شبکه ..... ۲۱
- ۳-۲-۲. پایدارسازی شبکه ..... ۲۲

## شکل‌ها

- شکل ۱- چند نمونه از نمونه‌های دیتاست ..... ۵
- شکل ۲- ساختار شبکه generator ..... ۵
- شکل ۳- ساختار شبکه discriminator ..... ۶
- شکل ۴- فریز کردن شبکه discriminator ..... ۶
- شکل ۵- آموزش شبکه‌ها ..... ۷
- شکل ۶- تصاویر تولیدی توسط شبکه مولد در مرحله ۱ آموزش ..... ۷
- شکل ۷- تصاویر تولیدی توسط شبکه مولد در مرحله ۶ آموزش ..... ۷
- شکل ۸- تصاویر تولیدی توسط شبکه مولد در مرحله ۲۱ آموزش ..... ۸
- شکل ۹- تصاویر تولیدی توسط شبکه مولد در مرحله ۴۱ آموزش ..... ۸
- شکل ۱۰- تصاویر تولیدی توسط شبکه مولد در مرحله ۴۶ آموزش ..... ۸
- شکل ۱۱- نمودار loss مدل ..... ۹
- شکل ۱۲- نمودار Accuracy مدل ..... ۹
- شکل ۱۳- روش smoothing ..... ۱۰
- شکل ۱۴- روش افزودن noise ..... ۱۰
- شکل ۱۵- تصاویر تولیدی توسط شبکه مولد پایدار در مرحله ۱ آموزش ..... ۱۰
- شکل ۱۶- تصاویر تولیدی توسط شبکه مولد پایدار در مرحله ۶ آموزش ..... ۱۱
- شکل ۱۷- تصاویر تولیدی توسط شبکه مولد پایدار در مرحله ۱۶ آموزش ..... ۱۱
- شکل ۱۸- تصاویر تولیدی توسط شبکه مولد پایدار در مراحل آخر آموزش ..... ۱۱
- شکل ۱۹- نمودار loss مدل پایداری‌سازی شده ..... ۱۲
- شکل ۲۰- نمودار Accuracy مدل پایداری‌سازی شده ..... ۱۲
- شکل ۲۱- تعریف شبکه discriminator طبقه بند ..... ۱۳
- شکل ۲۲- تعریف شبکه generator طبقه بند ..... ۱۳
- شکل ۲۳- ترکیب دو شبکه generator و discriminator طبقه بند ..... ۱۳
- شکل ۲۴- آموزش مدل طبقه‌بند ..... ۱۴
- شکل ۲۵- تصاویر تولیدی توسط شبکه مولد طبقه‌بند در مرحله ۱۰ آموزش ..... ۱۴
- شکل ۲۶- تصاویر تولیدی توسط شبکه مولد طبقه‌بند در مرحله ۳۰ آموزش ..... ۱۴
- شکل ۲۷- تصاویر تولیدی توسط شبکه مولد طبقه‌بند در مرحله ۵۰ آموزش ..... ۱۵

- شکل ۲۸- نمودار loss مدل تفکیک کننده طبقه بند ..... ۱۵
- شکل ۲۹- نمودار Accuracy مدل تفکیک کننده طبقه بند در طول آموزش ..... ۱۶
- شکل ۳۰- نمودار خطا و دقت مدل generator طبقه بند ..... ۱۶
- شکل ۳۱- تابع افزودن noise ..... ۱۷
- شکل ۳۲- تصاویر تولیدی توسط شبکه مولد طبقه بند پایدار سازی شده در مرحله ۱۰ آموزش ..... ۱۷
- شکل ۳۳- تصاویر تولیدی توسط شبکه مولد طبقه بند پایدار سازی شده در مرحله ۳۰ آموزش ..... ۱۷
- شکل ۳۴- تصاویر تولیدی توسط شبکه مولد طبقه بند پایدار سازی شده در مرحله ۵۰ آموزش ..... ۱۸
- شکل ۳۵- نمودار loss مدل discriminator طبقه بند پایدار سازی شده ..... ۱۸
- شکل ۳۶- نمودار loss مدل discriminator طبقه بند پایدار سازی شده ..... ۱۹
- شکل ۳۷- نمودار Accuracy و loss شبکه generator طبقه بند پایدار سازی شده در طول آموزش ..... ۱۹
- شکل ۳۸- Wasserstein loss function ..... ۲۰
- شکل ۳۹- اعمال تابع هزینه جدید ..... ۲۰
- شکل ۴۰- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۱ آموزش ..... ۲۰
- شکل ۴۱- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۱۱ آموزش ..... ۲۱
- شکل ۴۲- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۲۶ آموزش ..... ۲۱
- شکل ۴۳- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۴۶ آموزش ..... ۲۱
- شکل ۴۴- نمودار خطا شبکه WGAN ..... ۲۲
- شکل ۴۵- اضافه کردن smoothing به برچسب کلاس مثبت ..... ۲۲
- شکل ۴۶- اضافه کردن نویز به برچسب داده ها در شبکه WGAN ..... ۲۲
- شکل ۴۷- تصاویر تولیدی توسط شبکه مولد WGAN پایدار سازی شده در مرحله ۱ آموزش ..... ۲۳
- شکل ۴۸- تصاویر تولیدی توسط شبکه مولد WGAN پایدار سازی شده در مرحله ۱۶ آموزش ..... ۲۳
- شکل ۴۹- تصاویر تولیدی توسط شبکه مولد WGAN پایدار سازی شده در مرحله ۴۶ آموزش ..... ۲۳
- شکل ۵۰- نمودار loss شبکه WGAN تغییر یافته در طول آموزش ..... ۲۴
- شکل ۵۱- نمودار Accuracy شبکه WGAN تغییر یافته در طول آموزش ..... ۲۴

جدولها

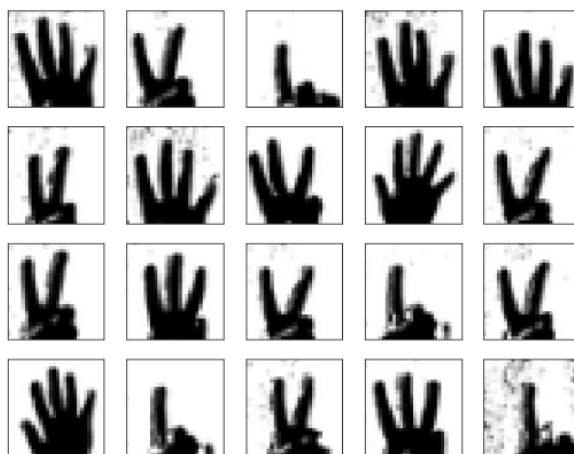
No table of figures entries found.

## پاسخ ۱. شبکه‌های مولد تخصصی کانولوشنال عمیق

### ۱-۱. پیاده‌سازی مولد تصویر با استفاده از شبکه‌های مولد تخصصی کانولوشنال

#### عمیق

در گام اول داده‌ها را در درایو گوگل قرار داده و آن‌ها را در کولب لود کردیم. سپس داده‌ها را به آموزش و ارزیابی تقسیم کرده و آن‌ها را نرمالایز کردیم.



شکل ۱- چند نمونه از نمونه‌های دیتاست

در این مرحله شبکه مولد را ساختیم. برای این کار از کتابخانه keras استفاده کردیم و یک شبکه sequential ساختیم. از آن جایی که تعداد داده‌ها زیاد نبود سعی کردیم شبکه پیچیدگی کمی داشته باشد و شبکه‌های با پارامترهای بزرگ نتایج خوبی به همراه نداشت.

این شبکه یک وکتور با سائز ۱۰۰ را به عنوان ورودی می‌گیرد. و در نهایت یک تصویر ساخته شده در سائز ۲۸ در ۲۸ در ۱ را خروجی می‌دهد.

```
num_features = 100

generator = keras.models.Sequential([
    keras.layers.Dense(7 * 7 * 128, input_shape=[num_features]),
    keras.layers.Reshape([7, 7, 128]),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2DTranspose(
        64, (5, 5), (2, 2), padding="same", activation="selu"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2DTranspose(
        1, (5, 5), (2, 2), padding="same", activation="tanh"),
])
```

شکل ۲- ساختار شبکه generator

سپس به ساختار شبکه تفکیک کننده پرداختیم. این شبکه تصویر ساخته شده توسط شبکه مولد را می گیرد و در خروجی باید مشخص کند آیا عکس واقعی است یا ساخته شده. به این دلیل که کلاس خروجی باینری است از sigmoid استفاده کردیم.

```
discriminator = keras.models.Sequential([
    keras.layers.Conv2D(64, (5, 5), (2, 2), padding = "same", input_shape = [28, 28, 1]),
    keras.layers.LeakyReLU(0.2),
    keras.layers.Dropout(0.3),
    keras.layers.Conv2D(128, (5, 5), (2, 2), padding = "same"),
    keras.layers.LeakyReLU(0.2),
    keras.layers.Dropout(0.3),
    keras.layers.Flatten(),
    keras.layers.Dense(1, activation = 'sigmoid')
])
discriminator.summary()
```

شکل ۳- ساختار شبکه discriminator

حالا نیاز است شبکه discriminator را compile کرده و آن را فریز کنیم. (به دلیل اینکه ابتدا باید شبکه مولد را آموزش دهیم) سپس شبکه را ترکیب می کنیم.

```
discriminator.compile(loss = "binary_crossentropy", optimizer = "adam")
discriminator.trainable = False
gan = keras.models.Sequential([generator, discriminator])

gan.compile(loss = "binary_crossentropy", optimizer = "adam")
```

شکل ۴ - فریز کردن شبکه discriminator

سپس با batch\_size=32 و ۵۰ مرحله به آموزش شبکه پرداختیم. در هر مرحله ابتدا نویز اولیه تولید می شود و آن را به شبکه مولد می دهیم تا تصویر غیر واقعی بسازد. سپس ترکیب تصاویر واقعی و غیر واقعی را به شبکه تفکیک کننده می دهیم تا آن را آموزش دهیم. و سپس به آموزش شبکه GAN می پردازیم به این صورت که وزن های شبکه تفکیک کننده را فریز کرده و نویز تصادفی تولید می کنیم و نویز را با لیبل یک به شبکه می دهیم.

```

seed = tf.random.normal(shape=[batch_size, 100])
D_loss=[]
D_acc=[]
G_acc=[]
G_loss=[]
def train_dcgan(gan, dataset, batch_size, num_features, epochs = 5):
    generator, discriminator = gan.layers
    for epoch in range(epochs):
        print()
        print("Epoch {}/{}".format(epoch + 1, epochs))

        for X_batch in dataset:
            # نویز 100 *
            # دادن نویز به شبکه مولد
            noise = tf.random.normal(shape=[batch_size, num_features])
            generated_images = generator(noise)

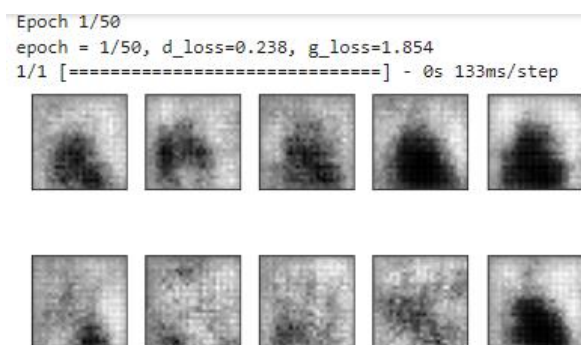
            # یک دسته از تصاویر واقعی و ساختگی می سازیم
            # از آن ها برای آموزش شبکه تفکیک کننده استفاده می کنیم
            X_fake_and_real = tf.concat([generated_images, X_batch], axis = 0)
            y1 = tf.constant([[0.]] * batch_size + [[1.]] * batch_size)
            discriminator.trainable = True
            d_loss,d_acc=discriminator.train_on_batch(X_fake_and_real, y1)
            D_loss.append(d_loss)
            D_acc.append(d_acc)

            # تصاویر ساخته شده توسط شبکه مولد را با لیبیل 1 به شبکه تفکیک کننده می دهیم
            noise = tf.random.normal(shape=[batch_size, num_features])
            y2 = tf.constant([[1.]] * batch_size)
            discriminator.trainable = False
            g_loss,g_acc=gan.train_on_batch(noise, y2)
            G_loss.append(g_loss)

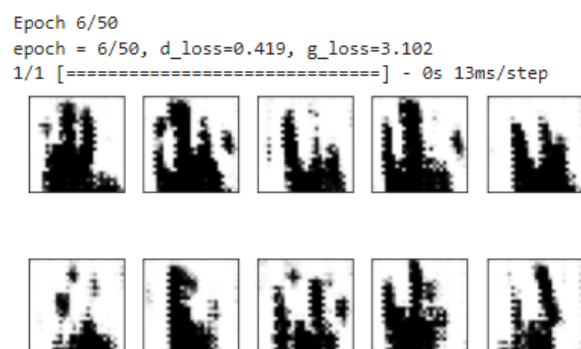
```

شکل ۵- آموزش شبکه‌ها

پیشرفت شبکه مولد در طول آموزش را در شکل‌های ۶ و ۷ و ۸ و ۹ و ۱۰ مشاهده می‌کنید.



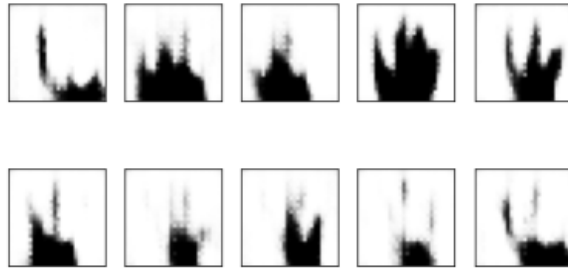
شکل ۶- تصاویر تولیدی توسط شبکه مولد در مرحله ۱ آموزش



شکل ۷- تصاویر تولیدی توسط شبکه مولد در مرحله ۶ آموزش



```
Epoch 21/50  
epoch = 21/50, d_loss=0.268, g_loss=3.830  
1/1 [=====] - 0s 13ms/step
```



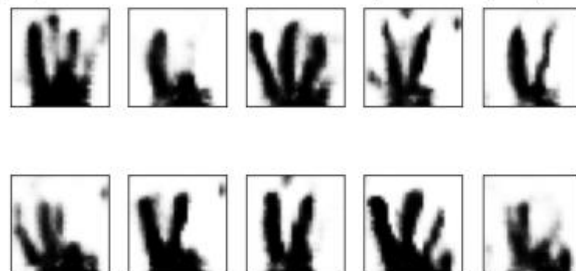
شکل ۸- تصاویر تولیدی توسط شبکه مولد در مرحله ۲۱ آموزش

```
Epoch 41/50  
epoch = 41/50, d_loss=0.293, g_loss=2.110  
1/1 [=====] - 0s 13ms/step
```



شکل ۹- تصاویر تولیدی توسط شبکه مولد در مرحله ۴۱ آموزش

```
Epoch 46/50  
epoch = 46/50, d_loss=0.252, g_loss=3.185  
1/1 [=====] - 0s 13ms/step
```



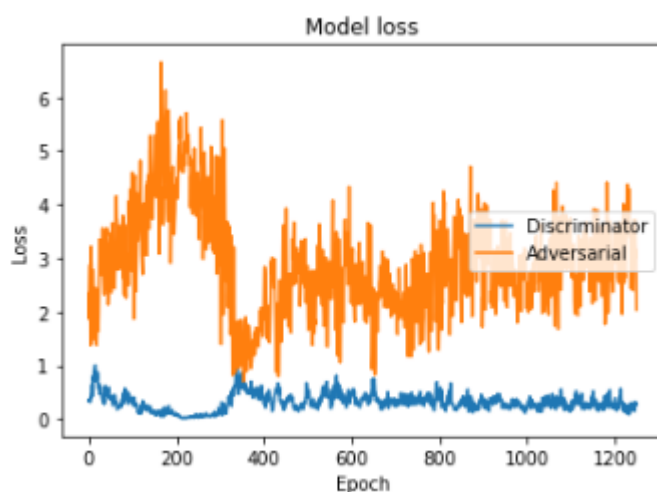
شکل ۱۰- تصاویر تولیدی توسط شبکه مولد در مرحله ۴۶ آموزش

همانطور که مشاهده کردید پیشرفت شبکه طی ایپاک‌ها مشهود است.

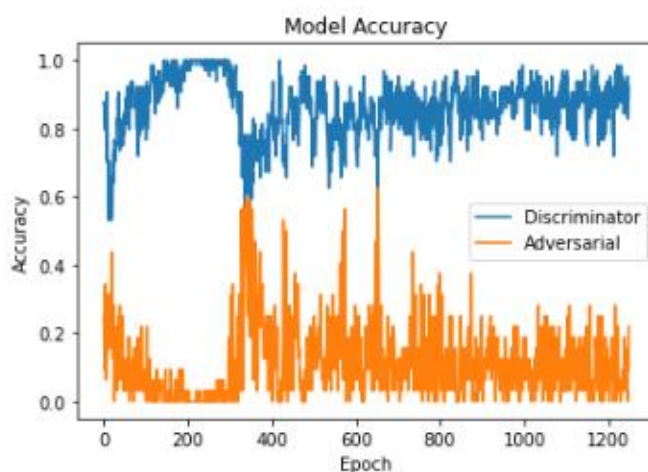
ما فکر میکنیم دقت بالای تفکیک کننده به این معنی نیست که تولید کننده با موفقیت متمایزکننده را فریب داده است یا خیر. در واقع، زمانی که دقت تشخیص‌دهنده نزدیک به ۵۰ درصد باشد، به این معنی است که مولد داده‌های ورودی را به خوبی مدل‌سازی می‌کند و تفکیک کننده نمی‌تواند بین این دو تفاوت قائل شود و حدس‌های تصادفی می‌زند.

## ۲-۱. ارزیابی شبکه

نمودار دقت و خطای شبکه نیز به شکل زیر بود.



شکل ۱۱- نمودار **loss** مدل



شکل ۱۲- نمودار **Accuracy** مدل

## ۳-۱. پایدارسازی شبکه

برای این کار از دو روش استفاده کردم. یکی **one-sided label smoothing** و دیگری **Adding noise**

روش اول :

استفاده از لیبل کلاس ۱ برای نمایش تصاویر واقعی و برچسب کلاس ۰ برای نمایش تصاویر جعلی در هنگام آموزش مدل تفکیک کننده معمول است. این برچسبها سخت نامیده می‌شوند

استفاده از برچسب های نرم، مانند مقادیر کمی بیشتر یا کمتر از ۱.۰ یا کمی بیشتر از ۰.۰ به ترتیب برای تصاویر واقعی و جعلی، که در آن مقدار تغییر برچسب برای هر تصویر تصادفی است، برای پایدار سازی شبکه بسیار مفید است. طبق گفته مقاله من فقط برای برچسب ۱ از این روش استفاده کردم.

```
y_fake=([[1.]] * batch_size)
y_fake_smooth= [[x[0] + -0.1 +(random() * 0.2)] for x in y_fake]
```

شکل ۱۳- روش smoothing

روش دوم:

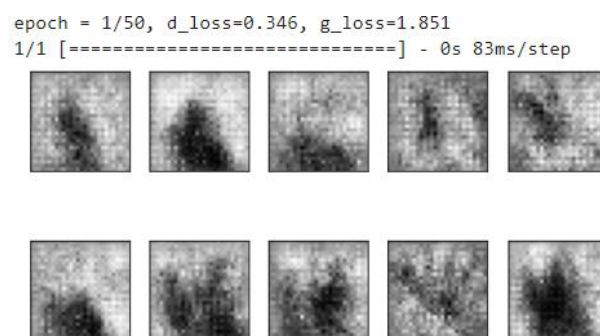
برچسب هایی که هنگام آموزش مدل تفکیک کننده استفاده می شوند، همیشه درست هستند.

توصیه می شود به این برچسب ها نویز هایی اضافه کنیم. به معنای افزودن تصادفی برخی از تصاویر جعلی به دسته ای از تصاویر واقعی یا افزودن تصادفی برخی از تصاویر واقعی به دسته ای از تصاویر جعلی با احتمال  $p\_flip$  می باشد.

```
def noisy_labels(p_flip, y):
    # determine the number of labels to flip
    n_select = int(p_flip * len(y))
    flip_ix = choice([i for i in range(len(y))], size=n_select)
    for i in flip_ix:
        y[i][0] = 1 - y[i][0]
        if(y[i][0]<0):
            y[i][0]=0
```

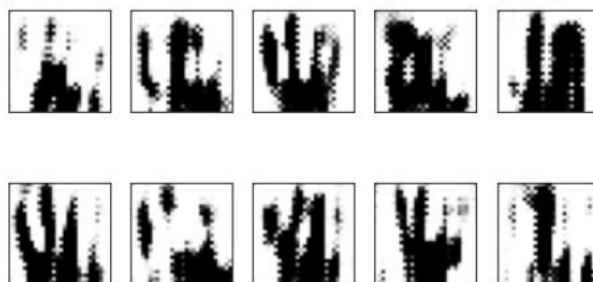
شکل ۱۴- روش افزودن noise

حالا با انجام تغییرات گفته شده دوباره مدل را برای ۵۰ مرحله آموزش دادیم. تصاویر تولیدی در طول آموزش به شرح زیر بود:



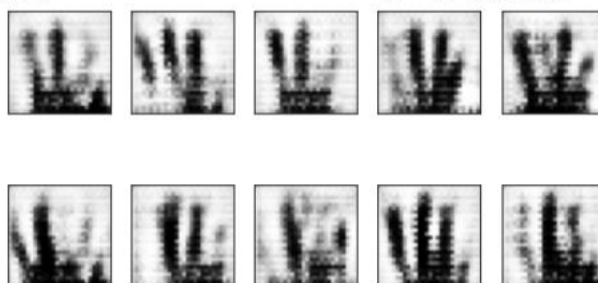
شکل ۱۵- تصاویر تولیدی توسط شبکه مولد پایدار در مرحله ۱ آموزش

epoch = 6/50, d\_loss=0.350, g\_loss=3.380  
1/1 [=====] - 0s 13ms/step



شکل ۱۶- تصاویر تولیدی توسط شبکه مولد پایدار در مرحله ۶ آموزش

epoch = 16/50, d\_loss=0.516, g\_loss=1.406  
1/1 [=====] - 0s 13ms/step



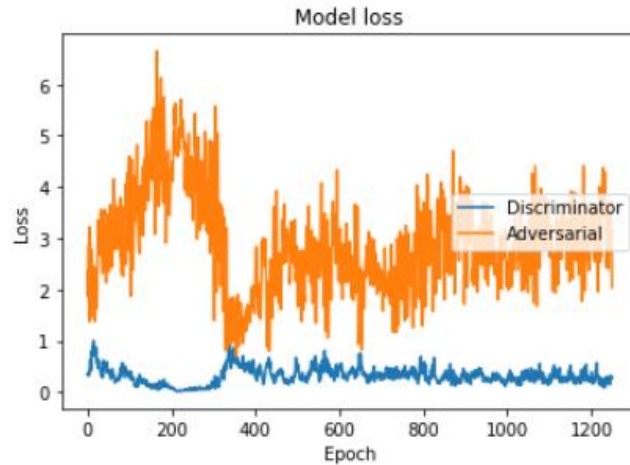
شکل ۱۷- تصاویر تولیدی توسط شبکه مولد پایدار در مرحله ۱۶ آموزش

epoch = 46/50, d\_loss=0.578, g\_loss=1.875  
1/1 [=====] - 0s 13ms/step

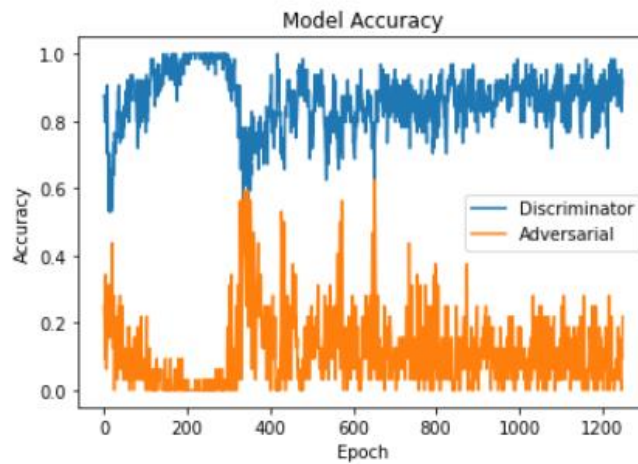


شکل ۱۸- تصاویر تولیدی توسط شبکه مولد پایدار در مراحل آخر آموزش

نمودار خطا و دقت مدل شبکه به صورت زیر می‌باشد:



شکل ۱۹- نمودار **loss** مدل پایداری سازی شده



شکل ۲۰- نمودار **Accuracy** مدل پایداری سازی شده

## پاسخ ۲. شبکه متخاصم مولد طبقه‌بند کمکی و شبکه Wasserstein

### ۲-۱. شبکه متخاصم مولد طبقه‌بند کمکی

#### ۲-۱-۱. پیاده‌سازی شبکه

در این مرحله در واقع قرار است پارامترهایی به مدل قبلی اضافه کنیم به نام کلاس تصویر به این معنی که شبکه علاوه بر واقعی بودن یا نبودن تصویر، کلاس آن را نیز پیش بینی کند. این کار وظیفه شبکه تفکیک کننده می باشد.

به این منظور برای شبکه تفکیک کننده ۲ خروجی تعریف کردم یکی برای برچسب واقعی بودن یا نبودن تصویر و دیگری برای برچسب کلاس تصویر، از همین رو از دو تابع loss نیز استفاده کردم.

```
# real/fake output
out1 = Dense(1, activation='sigmoid')(fe)
# class label output
out2 = Dense(n_classes, activation='softmax')(fe)
model = Model(in_image, [out1, out2])
opt = Adam(learning_rate=0.0002, beta_1=0.5)
model.compile(loss=['binary_crossentropy', 'sparse_categorical_crossentropy'],
return model
```

شکل ۲۱- تعریف شبکه **discriminator** طبقه بند

در شبکه مولد نیز یک امبدینگ برای بردار برچسب کلاس ها ساختم یک شبکه کوچک برای تولید این بردار و یک شبکه مثل شبکه مولد قبلی برای ساخت تصویر ساختم و آن ها را ترکیب کردم.

```
in_label = Input(shape=(1,))
# embedding for categorical input
li = Embedding(n_classes, 50)(in_label)
n_nodes = 7 * 7
li = Dense(n_nodes, kernel_initializer=init)(li)
li = Reshape((7, 7, 1))(li)
in_lat = Input(shape=(latent_dim,))
n_nodes = 384 * 7 * 7
gen = Dense(n_nodes, kernel_initializer=init)(in_lat)
gen = Activation('relu')(gen)
gen = Reshape((7, 7, 384))(gen)
# merge image gen and label input
merge = Concatenate()([gen, li])
```

شکل ۲۲- تعریف شبکه **generator** طبقه بند

حالا دو شبکه مولد و تفکیک کننده را ترکیب می کنم. این شبکه به عنوان ورودی برچسب و تصویر را می گیرد و در خروجی برچسب کلاس و برچسب واقعی بودن یا نبودن را می دهد.

```
# ترکیب مولد و تفکیک کننده
def define_gan(g_model, d_model):
    for layer in d_model.layers:
        if not isinstance(layer, BatchNormalization):
            layer.trainable = False
    gan_output = d_model(g_model.output)
    model = Model(g_model.input, gan_output)
    opt = Adam(learning_rate=0.0002, beta_1=0.5)
    model.compile(loss=['binary_crossentropy', 'sparse_categorical_crossentropy'],
return model
```

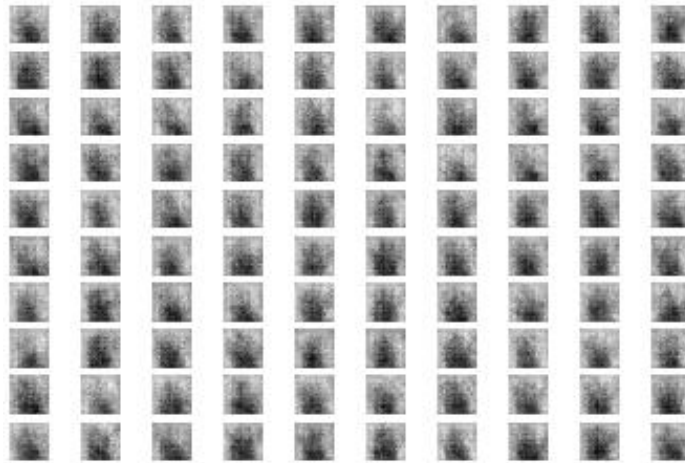
شکل ۲۳- ترکیب دو شبکه **generator** و **discriminator** طبقه بند

حالا مدل را برای ۵۰ مرحله آموزش می دهم (توضیحات چگونگی ترتیب فریز کردن شبکه ها برای آموزش در سوال قبلی گفته شده است).

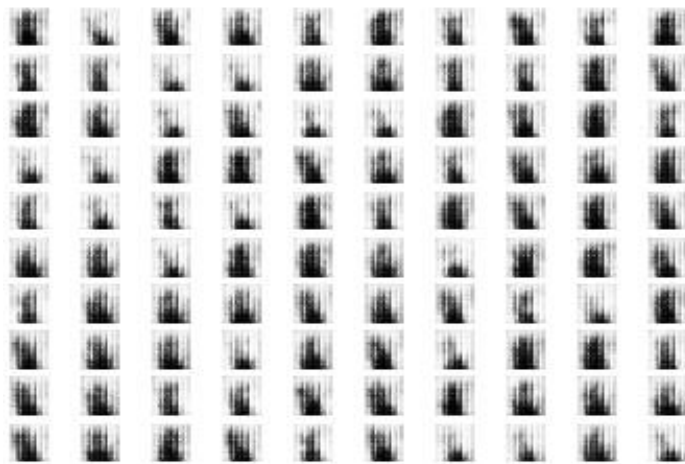
```
def train(g_model, d_model, gan_model, dataset, latent_dim, n_epochs=50, n_batch=64):
    d_loss=[]
    d_loss_labels=[]
    d_acc=[]
    g_loss=[]
    g_acc=[]
    d_acc_labels=[]
    bat_per_epo = int(dataset[0].shape[0] / n_batch)
    # calculate the number of training iterations
    n_steps = bat_per_epo * n_epochs
    half_batch = int(n_batch / 2)
    for i in range(n_steps):
        # get randomly selected 'real' samples
        [X_real, labels_real], y_real = generate_real_samples(dataset, half_batch)
        # آموزش مدل تفکیک کننده با داده های واقعی
        _,d_r1,d_r2,d_acc_r1,d_acc_r2 = d_model.train_on_batch(X_real, [y_real, labels_real])
        [X_fake, labels_fake], y_fake = generate_fake_samples(g_model, latent_dim, half_batch)
        # آموزش مدل تفکیک کننده با داده های فیک
        _,d_f,d_f2,d_acc_f1,d_acc_f2 = d_model.train_on_batch(X_fake, [y_fake, labels_fake])
        d_loss.append((d_f+d_r1)/2)
        d_loss_labels.append((d_f2+d_r2)/2)
```

شکل ۲۴- آموزش مدل طبقه‌بند

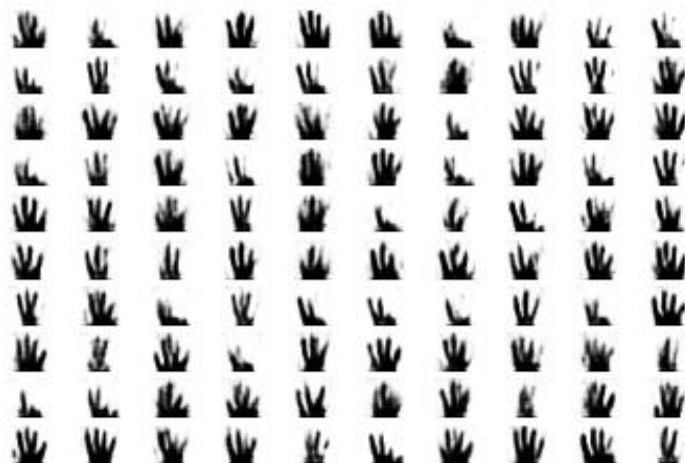
نتایج در طول آموزش بدست آمد. ابتدا تصاویر تولیدی توسط این شبکه:



شکل ۲۵- تصاویر تولیدی توسط شبکه مولد طبقه بند در مرحله ۱۰ آموزش



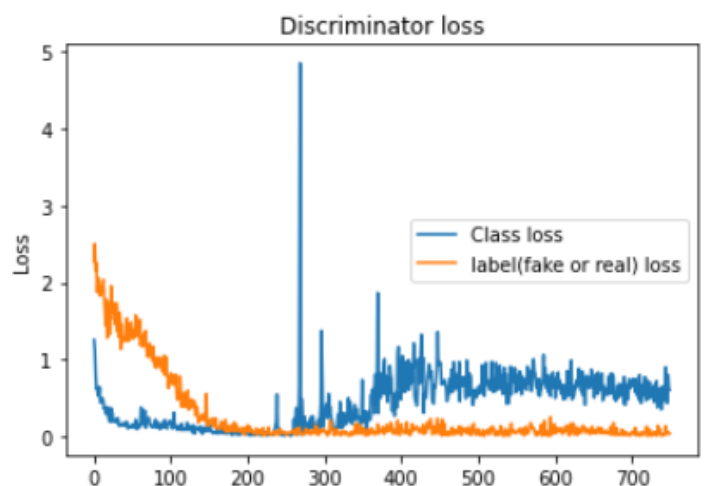
شکل ۲۶- تصاویر تولیدی توسط شبکه مولد طبقه بند در مرحله ۳۰ آموزش



شکل ۲۷- تصاویر تولیدی توسط شبکه مولد طبقه بند در مرحله ۵۰ آموزش

## ۲-۱-۲. ارزیابی شبکه

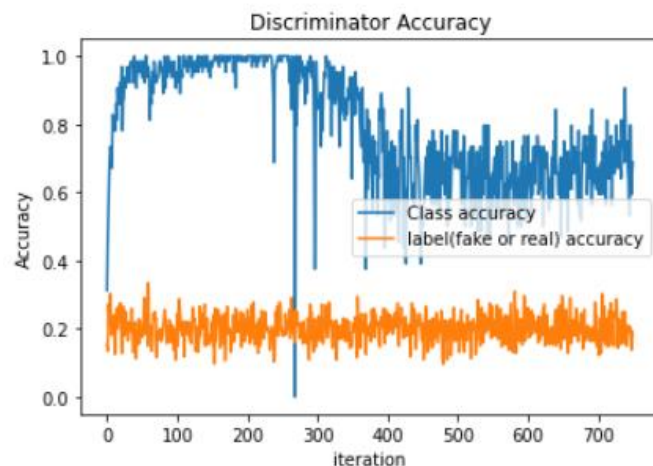
نمودار دقت و خطای مدل generator در طول آموزش را در شکل‌های ۲۸ و ۲۹ مشاهده می‌کنید.



شکل ۲۸- نمودار **loss** مدل تفکیک کننده طبقه بند

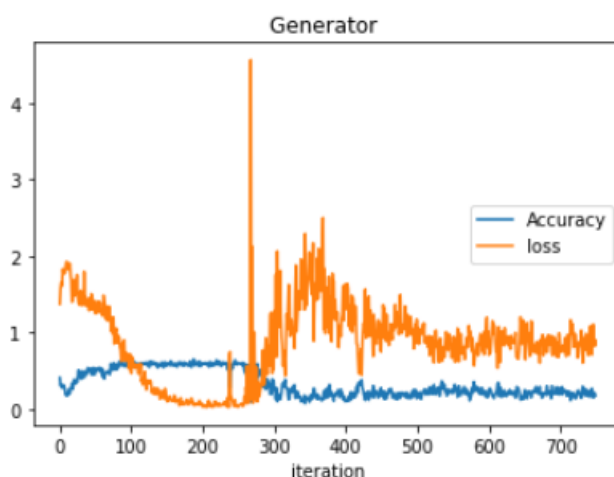
همانطور که مشاهده می‌کنید خطوط آبی خطا مدل برای تشخیص کلاس تصویر می‌باشد و خطوط نارنجی خطا مدل برای تشخیص واقعی یا فیک بودن تصویر می‌باشد که به خوبی همگرا شده است.





شکل ۲۹- نمودار **Accuracy** مدل تفکیک کننده طبقه بند در طول آموزش

نمودار دقت و خطای مدل discriminator در طول آموزش را در شکل ۳۰ مشاهده می کنید.



شکل ۳۰- نمودار خطا و دقت مدل **generator** طبقه بند

## ۳-۱-۲. پایدارسازی شبکه

در این مرحله برای پایدار سازی شبکه متخاصم طبقه بند تکنیک های گفته شده در سوال قبل را اجرا کردیم و مدل را برای ۵۰ مرحله آموزش دادیم:

```

from numpy.random import choice
def noisy_labels(p_flip, y):
    # determine the number of labels to flip
    n_select = int(p_flip * len(y))
    flip_ix = choice([i for i in range(len(y))], size=n_select)
    for i in flip_ix:
        y[i][0] = 1 - y[i][0]
        if(y[i][0]<0.1):
            y[i][0]=0
    return y

```

شکل ۳۱- تابع افزودن noise

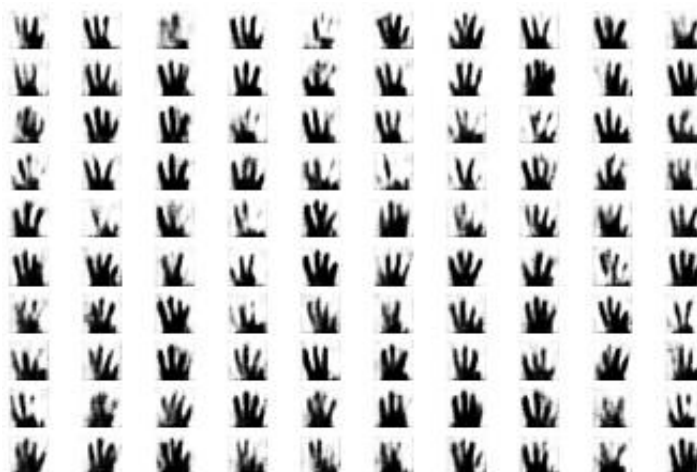
نتایج مدل در طول ۵۰ مرحله آموزش به شرح زیر می باشد:



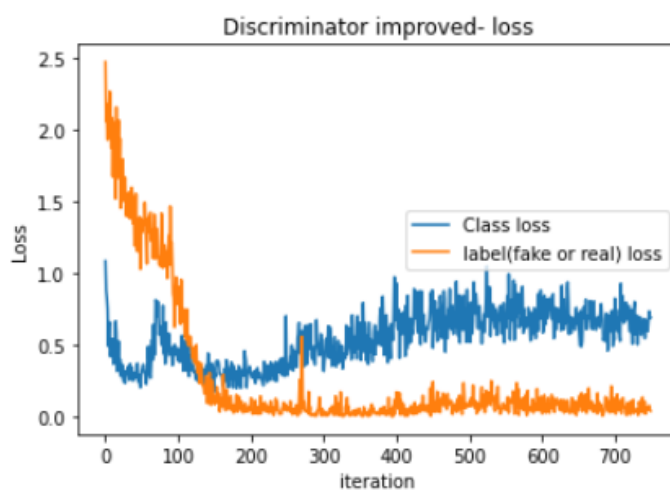
شکل ۳۲- تصاویر تولیدی توسط شبکه مولد طبقه بند پایدار سازی شده در مرحله ۱۰ آموزش



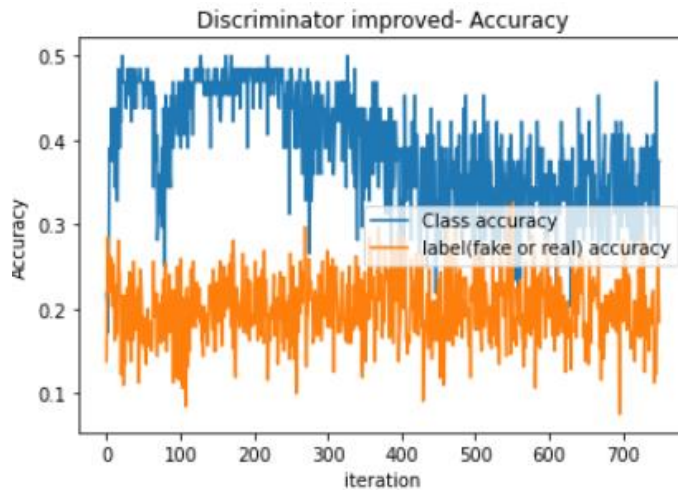
شکل ۳۳- تصاویر تولیدی توسط شبکه مولد طبقه بند پایدار سازی شده در مرحله ۳۰ آموزش



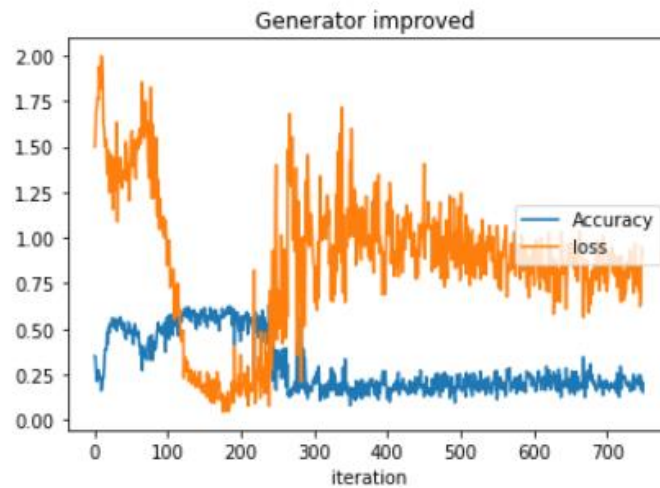
شکل ۳۴- تصاویر تولیدی توسط شبکه مولد طبقه بند پایدار سازی شده در مرحله ۵۰ آموزش  
 نمودار دقت و خطا شبکه پایدار سازی شده در طول آموزش را در شکل‌های ۳۵ و ۳۶ و ۳۷ مشاهده می‌کنید:



شکل ۳۵- نمودار **loss** مدل **discriminator** طبقه بند پایدار سازی شده



شکل ۳۶- نمودار **loss** مدل **discriminator** طبقه بند پایدار سازی شده



شکل ۳۷- نمودار **Accuracy** و **loss** شبکه **generator** طبقه بند پایدار سازی شده در طول آموزش همانطور که نمودار ها نیز مشهود است شبکه بهتر و پایدار تر شده و نوسان های کمتری در نتایج دارد.

## ۲-۲. شبکه متخاصم مولد Wasserstein

Wasserstein یکی از توابع هزینه است که بر اساس فاصله زمین متحرک (EMD) بین توزیع داده‌های تولید شده و داده‌های واقعی است. در حقیقت این روش، روشی برای اندازه گیری شباهت میان دو توزیع احتمال است.

روش جدید معرفی شده در این الگوریتم توانایی پیدا کردن فاصله ی نقاط در توزیع احتمال را با استفاده از فاصله ی موجود در تصاویر دیتاست دارد. بدین صورت شبکه قادر به یادگیری تا رسیدن به همگرایی می شود که در نتیجه ی آن، تصاویری با کیفیت بالاتر نمونه های تولیدی توسط مولد را شاهد خواهیم بود.

در مدل WGAN شبکه تفکیک کننده به جای اینکه احتمال واقعی بودن یک تصویر تولید شده را پیش بینی کند به واقعی بودن یک تصویر معین احتمالی را نسبت می دهد.

Critic Loss = [average critic score on real images] – [average critic score on fake images]

Generator Loss = -[average critic score on fake images]

## ۱-۲-۲. پیاده سازی شبکه

برای پیاده سازی این تابع هزینه از تابع زیر استفاده کردم:

```
import keras.backend as K
def wasserstein_loss(y_true, y_pred):
    return K.mean(y_true * y_pred)
```

شکل ۳۸- Wasserstein loss function

و مدل GAN نوشته شده در سوال یک را به این صورت تغییر دادم:

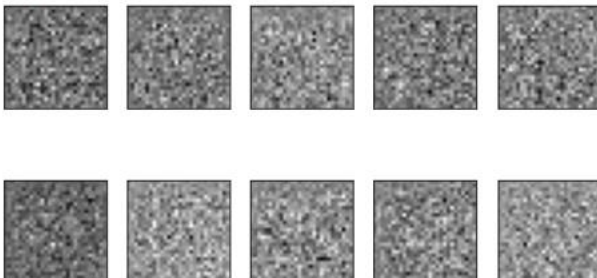
```
discriminator.compile(loss =wasserstein_loss, optimizer ="adam", metrics=['accuracy'])
discriminator.trainable = False
gan = keras.models.Sequential([generator, discriminator])

gan.compile(loss =wasserstein_loss, optimizer ="adam" ,metrics=['accuracy'])
```

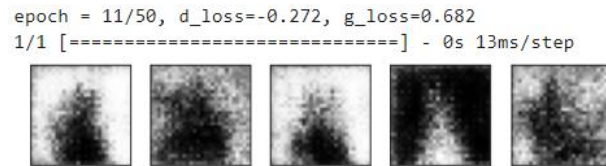
شکل ۳۹- اعمال تابع هزینه جدید

نکته مهم دیگر این است که در این مدل برای محاسبه تابع هزینه جدید برچسب کلاس واقعی به جای ۰ ، ۱ می باشد. مدل را برای ۵۰ مرحله (مانند سوال های قبلی) آموزش دادم. تصاویر تولید شده توسط شبکه مولد در طول آموزش به شکل زیر بود.

```
epoch = 1/50, d_loss=-0.167, g_loss=0.367
1/1 [=====] - 0s 79ms/step
```



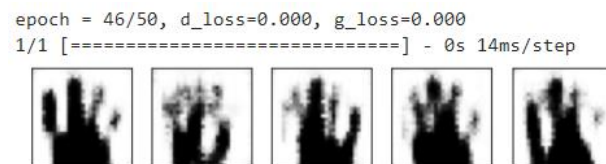
شکل ۴۰- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۱ آموزش



شکل ۴۱- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۱۱ آموزش



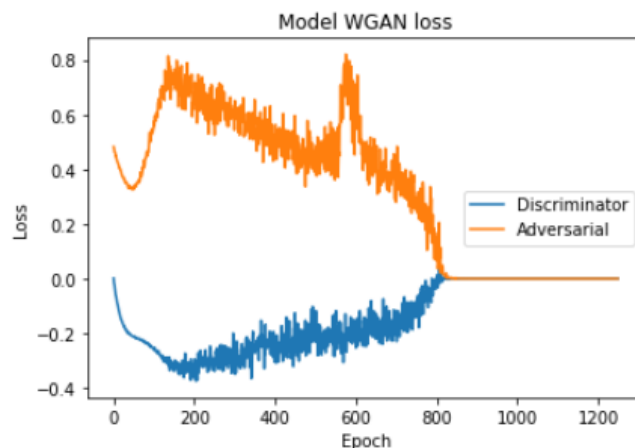
شکل ۴۲- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۲۶ آموزش



شکل ۴۳- تصاویر تولیدی توسط شبکه مولد WGAN در مرحله ۴۶ آموزش

## ۲-۲-۲. ارزیابی شبکه

نمودار خطای این مدل را در شکل ۴۴ مشاهده می کنید.



شکل ۴۴- نمودار خطا شبکه WGAN

همانطور که در شکل ۴۴ مشخص است خطا برای هر دو مدل کم و همگرا شده است.

### ۳-۲-۲. پایدارسازی شبکه

در این مرحله برای پایدار سازی شبکه روش های گفته شده در سوال یک را روی این شبکه اعمال کردم. پیاده سازی روش های گفته شده تغییر زیادی ندارد فقط باید در قسمت add noise با توجه به ۱- بودن برچسب تصاویر واقعی تغییراتی اعمال شود.

```
y_fake_smooth= [[x[0] + -0.1 +(random() * 0.2)] for x in y_fake]
```

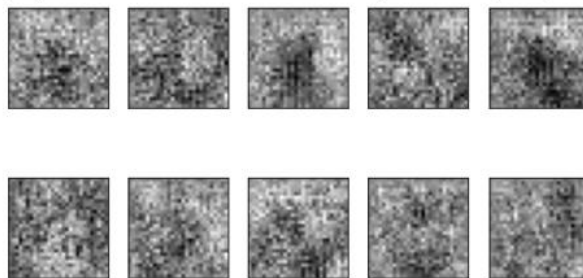
شکل ۴۵- اضافه کردن smoothing به برچسب کلاس مثبت

```
def noisy_labels(p_flip, y):
    # determine the number of labels to flip
    n_select = int(p_flip * len(y))
    flip_ix = choice([i for i in range(len(y))], size=n_select)
    for i in flip_ix:
        y[i][0]= 0 - y[i][0]
        if(y[i][0]<0):
            y[i][0]=-1
    return y
```

شکل ۴۶- اضافه کردن نویز به برچسب داده ها در شبکه WGAN

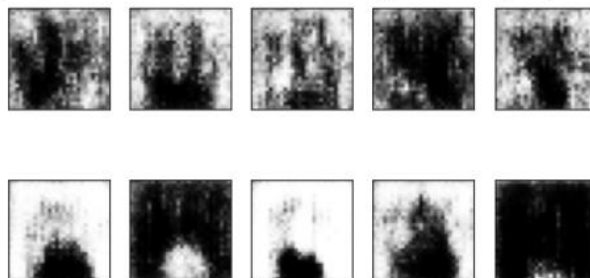
شبکه تغییر یافته را برای ۵۰ مرحله آموزش دادم. تصاویر تولیدی توسط شبکه مولد به شکل زیر بود.

```
epoch = 6/50, d_loss=-0.307, g_loss=0.695  
1/1 [=====] - 0s 21ms/step
```



شکل ۴۷- تصاویر تولیدی توسط شبکه مولد WGAN پایدار سازی شده در مرحله ۱ آموزش

```
epoch = 16/50, d_loss=-0.220, g_loss=0.449  
1/1 [=====] - 0s 14ms/step
```



شکل ۴۸- تصاویر تولیدی توسط شبکه مولد WGAN پایدار سازی شده در مرحله ۱۶ آموزش

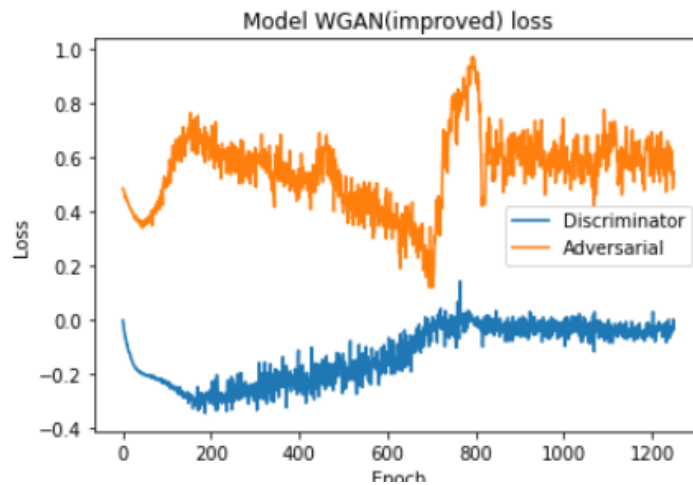
```
epoch = 46/50, d_loss=-0.037, g_loss=0.603  
1/1 [=====] - 0s 20ms/step
```



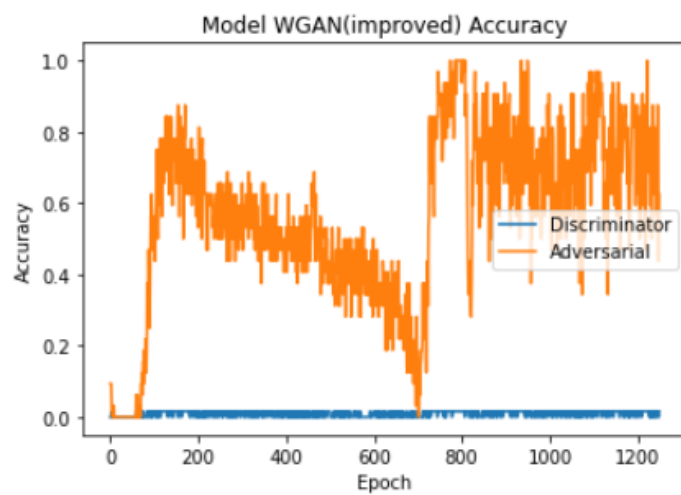
شکل ۴۹- تصاویر تولیدی توسط شبکه مولد WGAN پایدار سازی شده در مرحله ۴۶ آموزش

نمودار loss و Accuracy شبکه WGAN تغییر یافته را در شکل‌های ۵۰ و ۵۱ مشاهده می‌کنید.





شکل ۵۰- نمودار **loss** شبکه **WGAN** تغییر یافته در طول آموزش



شکل ۵۱- نمودار **Accuracy** شبکه **WGAN** تغییر یافته در طول آموزش

بنظر می رسد روش های معرفی شده روی این شبکه کارایی لازم را ندارد.