دانشگاه شهید بهشتی

دانشکده علوم ریاضی

$autoencoders$

تمرین سوم درس شبکه عصبی

نگارش

زهرا موسی خانی

استاد راهنما

دکتر سعیدرضا خردپیشه

بهار ۱۴۰۰

فهرست مطالب

فهرست شکل‌ها

فصل ۱

تمرین اول

دیتاست این مسئله شامل ۲۰۲۴ عکس از نقاشی ها می باشد و هدف از این تمرین رنگی کردن این عکس ها می باشد. به این ترتیب که ابتدا این عکس های داده شده را به عکس های سیاه و سفید تبدیل کنیم. سپس با استفاده از اتوانکودرها این عکس های سیاه و سفیده شده را رنگی کنیم. در ابتدا نیز متوجه شدیم فرمت چند تا از این عکس ها به صورتی بود که قابل خوانده شدن نبود درنتیجه ابتدا به حذف این تعداد داده پرداختیم که تعداد آنها خیلی هم زیاد نبود. نمونه ای از عکس های این دیتاست در شکل شماره ۱ قابل مشاهده می باشد. همان طور که در شکل نیز مشاهده می شود، این عکس ها دارای ابعاد یکسانی نیستند که برای راحتی کار ابعاد آنها را یکسان کردیم. حال در ادامه روند کار را مرحله به مرحله توضیح میدهیم.

سیاه سفید کردن عکس ها

ابتدا عکس ها را بااستفاده از تابع $rgb2gray$ به عکس های سیاه سفید تیدیل کردیم و در فولدر جدیدی عکسای سیاه سفید و اصلی را قرار دادیم و با استفاده از دیتالودر و بچ سایز ۶۴ داده ها را برای آموزش شبکه آماده سازی کردیم. نمونه ای از عکس های سیاه و سفید شده را در شکل شماره ۲ میتوانید مشاهده کنید.

تابع خطا مناسب

با بررسی مقالات و منابع موجود در اینترنت یافتیم که استفاده از تابع هزینه $mean\ squared\ error$ باعث می شود تا خروجی رنگی مابین رنگ های قرمز، آبی و سیز داشته باشد و در واقع معادل قهوه ای باشد. از آنجاییکه

شکل ۱.۱: شکل اول : نمونه داده ها

شکل ۲.۱: شکل دوم : نمونه عکس های سیاه و سفید

بیشتر عکس های ما در این دیتاست نیز شامل همین رنگ است این تاع خطا را برای مدل خود انتخاب کردیم. حال به بررسی ساختار شبکه منتخب خود و تاثیر آپتیمایزر و لرنینگ ریت های مختلف و همچنین عمق شبکه میپردازیم. ابتدا ساختار مدل منتخبی که استفاده کردیم را ارائه می دهیم. این ساختار به صورت جزئی در شکل شماره ۳ آمده است. حال به بررسی نتایج با استفاده از آپتیمایزر های مختلف، لرنینگ ریت مختلف و عمق شبکه میپردازیم. در ابتدا از آپتیمایزر $Adam$ با لرنینگ ریت ۰۱.۰ استفاده کردیم و به تعداد ۱۳۰ ایپاک برروی مجموعه داده آموزش دادیم. چند نمونه از نتایج حاصل را طی ایپاک های مختلف و روند رنگی شدن عکسا ها را در شکل شماره ۴ مشاهده میکنیم. سمت راست ترین عکس در این نتایج عکس اصلی می باشد و در بقیه عکس ها روند رنگی شدن عکس را طی چند ایپاک مختلف تا ایپاک نهایی که ۱۳۰ می باشد مشاهده میکنیم.

مدل دوم

این بار باز هم از همان آپتیمایزر $Adam$ استفاده کردیم با این تفاوت که مقدار لرنینگ ریت را برابر ۰۴.۰ در نظر گرفتیم. برای اینکه نتایج با مدل قبل قابل مقایسه باشد، در شماره ایپاک های مشابه مدل قبل نتایج را قرار

```
ColorizationNet(
  (encoder): Sequential(
    (0): Conv2d(1, 12, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(12, 32, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): ConvTranspose2d(32, 12, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): ConvTranspose2d(12, 2, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
    (7): ReLU()
  )
)
```

شکل ۳.۱: شکل سوم : ساختار شبکه

میدهیم و این نتایج در شکل شماره ۵ قابل مشاهده می باشند. همان طور که در تصویر شماره ۵ مشاهده می شود، تفاوت خیلی چشم گیری بین نتایج حاصل نمیشود اما با مقایسه با نتایج مدل قبل متوجه میشویم که به نظر مدل قبل در تمایز رنگ ها و بولد کردن قسمت های روشن و تیره عکس به خصوص در ایپاک های اولیه نسبت به این مدل بهتر عمل می کند. حال مدل را کمی عمیق تر میکنیم و به بررسی نتایج میپردازیم. ابتدا قصد داشتیم مدل را خیلی عمیق کنیم اما مدام به ارور حافظه در کولب برخورد میکردیم تا درنهایت توانستیم از مدل ارائه شده در شکل ۶ اجرا بگیریم.

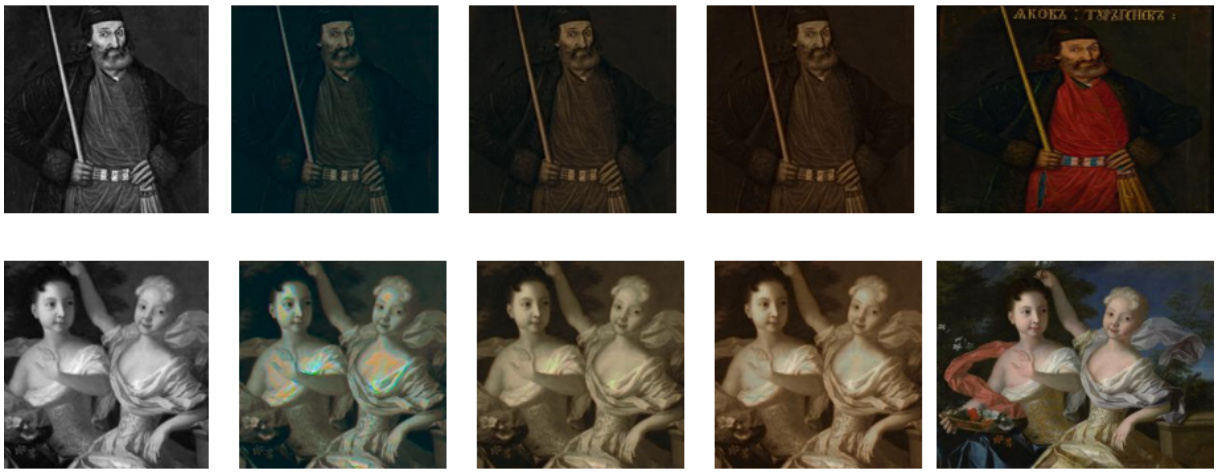
مدل سوم

حال این مدل را با استفاده از آپتیمایزر $Adam$ و لرنینگ ریت ۰۱.۰ امتحان میکنیم و نتیایج به همراه عکس اصلی را می توان در شکل شماره ۷ مشاهده نمود.

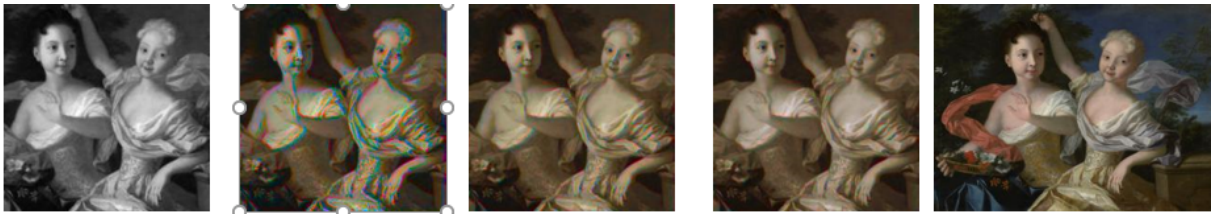
مدل چهارم

حال دوباره از همان مدل ساده اول استفاده میکنیم و این بار از آپتیمایزر $RMSprop$ با لرنینگ ریت ۰۱.۰ استفاده کرده و نتایج در شکل ۸ قابل مشاهده می باشد.

مدل پنجم در این مرحله از همان ساختار شبکه و آپتیمایزر $RMSprop$ استفاده میکنیم اما لرنینگ ریت را به ۰۰۵.۰ تغییر میدهیم وچ ند نمونه از نتایج حاصل از این مدل را می توان در تصویر شماره ۹ مشاهده کرد. همان



شکل ۴.۱: شکل چهارم : نتایج حاصل از مدل اول



شکل ۵.۱: شکل پنجم : نتایج حاصل از مدل دوم

طور که در شکل هشتم مشاهده میشود این تغییر لرنینگ ریت باعث می شود نتایج خوبی حاصل نشود و نسبت به مدل قبل عکسا تیره تر هستند و به خوبی رنگ های تیره و روشن تفکیک نشده اند.

مدل ششم

این بار با همان مدل و از آپتیمایزر SGD با لرنینگ ریت ۰۱.۰ و $momentum$ ۵.۰ استفاده میکنیم. چند نمونه از نتایج حاصل از این مدل را در شکل شماره ۹ میتوان دید. لازم به ذکر است در تمامی این مدل ها به جز مواردی که ذکر شد، در بقیه موارد تعداد ایپاک ها ۲۰۰ می باشد.

مدل هفتم

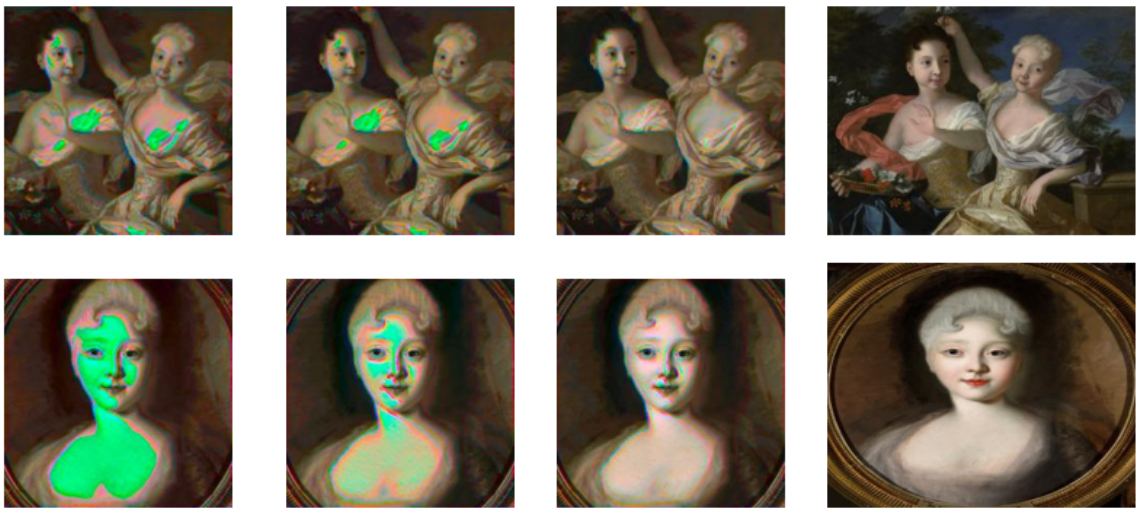
در این مدل، از شبکه $pre\ train\ resnet$ برای بخش انکودر استفاده کردیم. از آپتیمایزر $Adam$ با لرنینگ ریت ۰۱.۰ و با توجه به اینکه محدودیت استفاده از GPU در کولب تمام شده بود، برای تعداد ۶۰ ایپاک کد را ران کردیم که ساختار مدل و نتایج حاصل از آن را می توان به ترتیب در شکل های ۱۰ و ۱۱ و ۱۲ به ترتیب مشاهده کرد. همان طور که در شکل شماره ۱۲ مشاهده میشود، عکس های رنگش ده به نسبت بهتر از عکس های مدل

```
ColorizationNetdeep(
  (encoder): Sequential(
    (0): Conv2d(1, 12, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(12, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU()
  )
  (decoder): Sequential(
    (0): Conv2d(64, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Conv2d(32, 12, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (7): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Conv2d(12, 2, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (10): ReLU()
  )
)
```

شکل ۶.۱: شکل ششم : مدل عمیق

های قبلی می باشند اما خیلی هم فرق چندانی با مدل های برتری که قبلا ارائه شد ندارند. اما این نکته را نیز باید درنظر بگیریم که این مدل با تعداد ایپاک ۶۰ به این نتایج رسیده است و این نکته بسیار قابل توجه می باشد.

شکل ۷.۱: شکل هفتم : نتایج مدل عمیق



شکل ۸.۱: شکل هشتم : نتایج مدل چهارم

شکل ۹.۱: شکل هشتم : نتایج مدل پنجم

شکل ۱۰.۱: شکل نهم : نتایج مدل ششم

```
ColorizationResNet(
  (midlevel_resnet): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (5): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
```

شکل ۱۱.۱: شکل دهم : مدل $resnst$

```
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
  (upsample): Sequential(
    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Upsample(scale_factor=2.0, mode=nearest)
    (4): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU()
    (10): Upsample(scale_factor=2.0, mode=nearest)
    (11): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (12): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (13): ReLU()
    (14): Conv2d(32, 2, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): Upsample(scale_factor=2.0, mode=nearest)
  )
)
```

شکل ۱۲.۱: شکل یازدهم : مدل $resnst$

شکل ۱۳.۱: شکل دوازدهم : نتایج مدل آخر

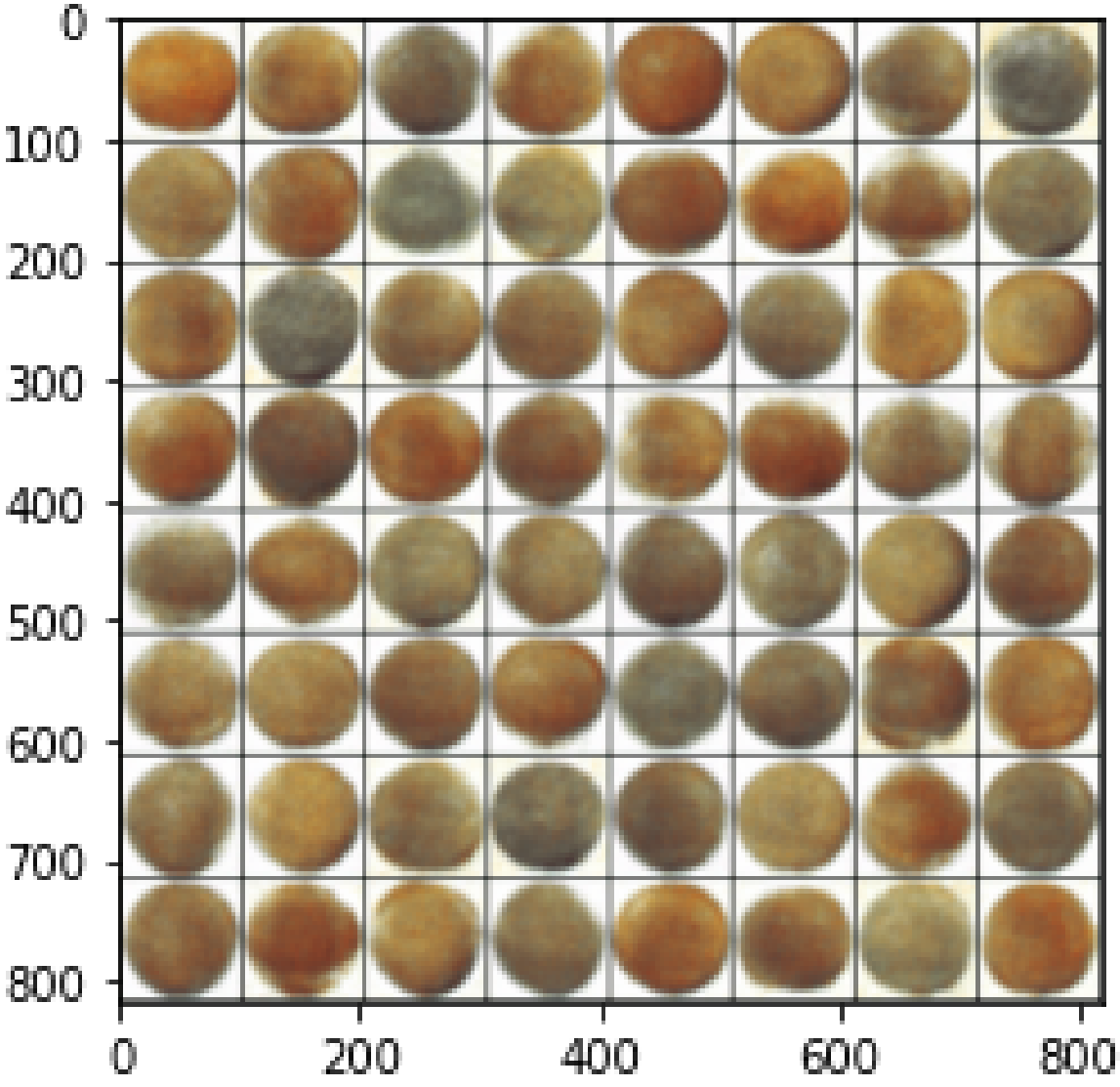
فصل ۲

تمرین دوم

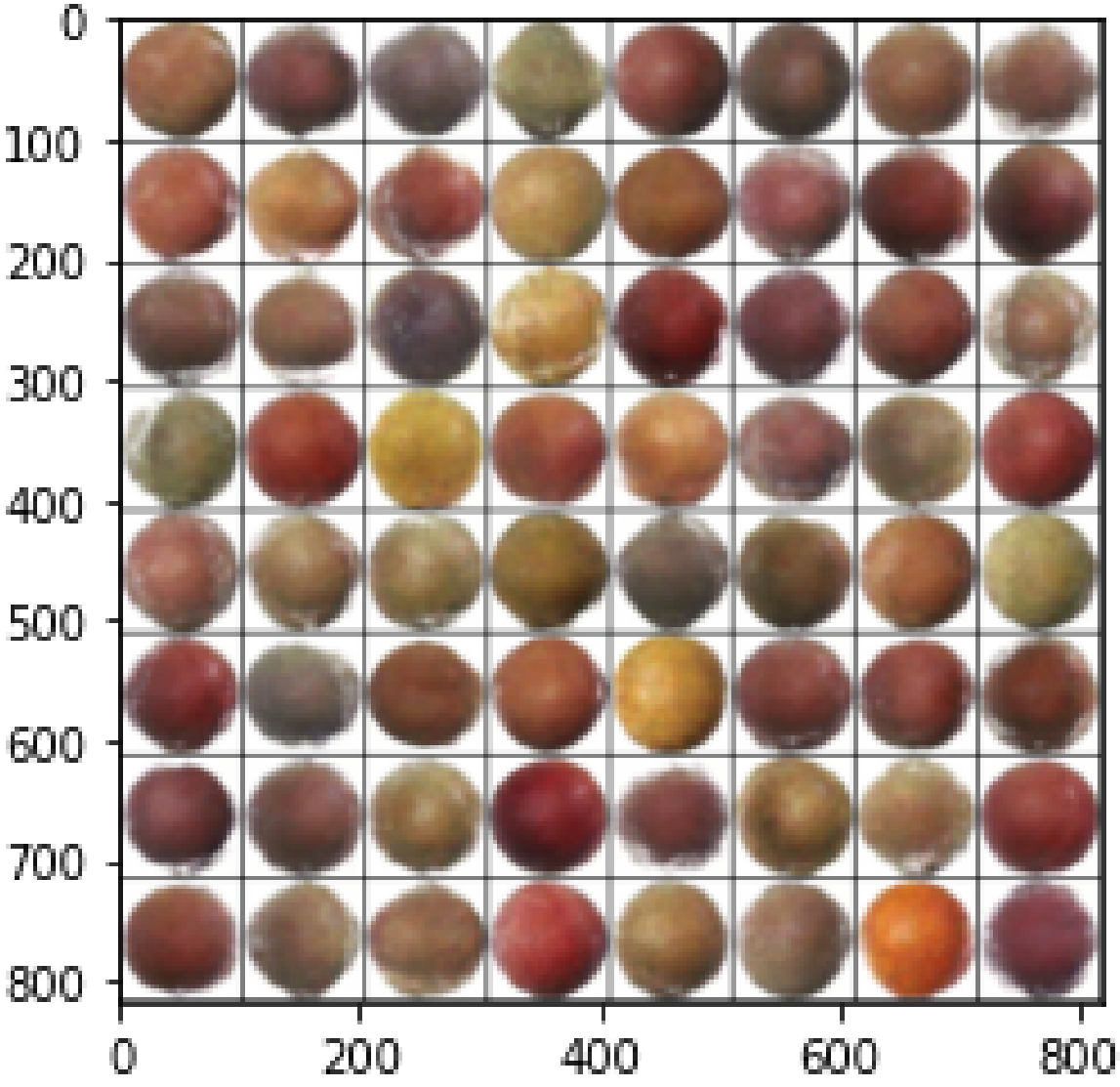
هدف این تمرین پیاده سازی یک $variational\ auto\ encoder$ می باشد. دیتاست شامل عکس های میوه ها می باشد و هدف این است که با استفاده از شبکه بتوانیم عکس های جدید تولید بکنیم. با استفاده از لینکی که در صورت سوال قرار داده شده بود به طراحی شبکه پرداختیم که جزئیات آن را میتوان در شکل شماره ۱۳ مشاهده کرد. ابتدا سعی کردیم برای تعداد اپیاک بالا شبکه را آموزش دهیم اما به دلیل حجم بالای داده ها و خوردن به محدودیت حافظه تنها توانستیم برای ۵ اپیاک این کار را انجام دهیم. حال اگر توزیع داده ها را به $decoder$ بدهیم امید داریم میوه های جدیدی تولید شوند نتایج را در شکل شماره ۱۴ مشاهده میکنیم. طبیعی است هرچه تعداد اپیاک ها بیشتر شود تصاویر بهتری تولید میشوند.

```
VAE(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(64, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=10000, out_features=2048, bias=True)
  (fc_bn1): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc21): Linear(in_features=2048, out_features=2048, bias=True)
  (fc22): Linear(in_features=2048, out_features=2048, bias=True)
  (fc3): Linear(in_features=2048, out_features=2048, bias=True)
  (fc_bn3): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc4): Linear(in_features=2048, out_features=10000, bias=True)
  (fc_bn4): BatchNorm1d(10000, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv5): ConvTranspose2d(16, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), output_padding=(1, 1), bias=False)
  (bn5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv6): ConvTranspose2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn6): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv7): ConvTranspose2d(32, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), output_padding=(1, 1), bias=False)
  (bn7): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv8): ConvTranspose2d(16, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (relu): ReLU()
)
```

شکل ۱.۲: شکل سیزدهم : ساختار شبکه



شکل ۲.۲: شکل چهاردهم : نتایج ایپاک اول



شکل ۳.۲: شکل پانزدهم : نتایج ایپاک پنجم