

Trabalho escolhido: [CuratorNet: Visually-aware Recommendation of Art Images | Papers With Code](#)

Aluno: Victor Accete Nicácio Placido

# Replicação

(Para a segunda fase do projeto, referente a AB2, conferir a seção [Tentativas de melhorias](#))

## Instalação dos pacotes necessários

No github os autores colocam algumas instruções, bem como os notebooks necessários para a reprodução do artigo. Tentei seguir as instruções de instalação, mas encontrei alguns percalços.

### To try it out

---

Execute the following commands from the main folder:

- `virtualenv -p python3 ./env`
- `source ./env/bin/activate`
- `pip install -r requirements.txt`
- `ipython kernel install --name "CuratorNetKernel" --user`
- `jupyter notebook`

No entanto, não foi possível apenas seguir as instruções diretamente por conta da minha versão do Python. Eu tinha Python 3.9 instalado na minha máquina e por isso recebia a seguinte mensagem de erro:

```
(base) C:\Users\victor_accete\CuratorNet>pip install -r requirements.txt
ERROR: Could not find a version that satisfies the requirement tensorflow==1.14.0
ERROR: No matching distribution found for tensorflow==1.14.0
```

Por ser uma versão antiga do TensorFlow, criei um ambiente com uma versão mais antiga do Python, a 3.6.2. Depois disso consegui usar corretamente o comando para instalar os pacotes em requirements.txt.

## Notebook de treinamento

As 9 primeiras células de código rodaram corretamente, mas aconteceu um erro na décima célula de código.

## Load clusters

```
In [10]: with open(data_path + 'clusters.json') as f:
          artworkId2clusterId = json.load(f)
          cluster_ids = np.full((n_artworks,), -1, dtype=int)
          for k, v in artworkId2clusterId.items():
              cluster_ids[artwork_id2index[k]] = v

-----
KeyError                                Traceback (most recent call last)
<ipython-input-10-0f424ed9cea6> in <module>
      3 cluster_ids = np.full((n_artworks,), -1, dtype=int)
      4 for k, v in artworkId2clusterId.items():
----> 5     cluster_ids[artwork_id2index[k]] = v

KeyError: '897c4dba06f7b4ff60f3c1db09e880d5'
```

No entanto, no repositório deles, nas issues, achei uma pessoa com um problema parecido que postou uma solução. Este é o link da issue:

<https://github.com/ialab-puc/CuratorNet/issues/1>.

## Treinamento

Com os passos anteriores, consegui rodar o código de treinamento.

```
Before training: test_accuracy = 0.489099
Starting training ...
EPOCH=1 => train_i=483, train_loss = 0.794118459809, test_accuracy = 0.7595909, epoch_secs = 69.61, total_secs = 69.61
** improvement detected: model saved to path curatornet_10m/
EPOCH=2 => train_i=483, train_loss = 0.868082020532, test_accuracy = 0.8540577, epoch_secs = 68.94, total_secs = 138.54
** improvement detected: model saved to path curatornet_10m/
EPOCH=3 => train_i=483, train_loss = 0.763863758910, test_accuracy = 0.8783451, epoch_secs = 69.41, total_secs = 207.95
** improvement detected: model saved to path curatornet_10m/
EPOCH=4 => train_i=483, train_loss = 0.641372515237, test_accuracy = 0.8941769, epoch_secs = 68.46, total_secs = 276.42
** improvement detected: model saved to path curatornet_10m/
EPOCH=5 => train_i=483, train_loss = 0.531042417020, test_accuracy = 0.9037714, epoch_secs = 69.82, total_secs = 346.24
** improvement detected: model saved to path curatornet_10m/
EPOCH=6 => train_i=483, train_loss = 0.439340349201, test_accuracy = 0.9101259, epoch_secs = 69.57, total_secs = 415.80
** improvement detected: model saved to path curatornet_10m/
EPOCH=7 => train_i=483, train_loss = 0.365543774611, test_accuracy = 0.9161104, epoch_secs = 154.20, total_secs = 570.00
** improvement detected: model saved to path curatornet_10m/
EPOCH=8 => train_i=483, train_loss = 0.306990775106, test_accuracy = 0.9210752, epoch_secs = 64.15, total_secs = 634.15
** improvement detected: model saved to path curatornet_10m/
EPOCH=9 => train_i=483, train_loss = 0.261265349413, test_accuracy = 0.9182546, epoch_secs = 57.28, total_secs = 691.43
EPOCH=10 => train_i=483, train_loss = 0.224611175584, test_accuracy = 0.9361279, epoch_secs = 4.18, total_secs = 695.60
** improvement detected: model saved to path curatornet_10m/
===== TIMEOUT =====
```

Rodei mais algumas vezes o notebook de treinamento, mas agora carregando o checkpoint.

Na terceira vez que rodei:

```
INFO:tensorflow:Restoring parameters from C:\U
model successfully restored from checkpoint!
Before training: test_accuracy = 0.927588
Starting training ...
```

Ao fim da terceira vez ficou assim:

```
test_accuracy = 0.971675
```

## Notebook de precomputação de embeddings

Novamente, mais um erro na antepenúltima célula deste notebook.

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-27-0b5056f7c847> in <module>
    11     with tf.Session(config=config) as sess:
    12         saver = tf.train.Saver()
--> 13         saver.restore(sess, tf.train.latest_checkpoint(MODEL_PATH))
    14         item_vectors = network.precompute_tensors(sess, concat_featmat)

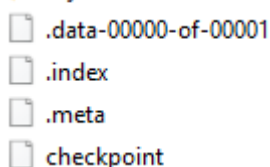
~\Anaconda3\envs\curatornet\lib\site-packages\tensorflow\python\tensor_util.py in restore(self, sess, save_path)
    1272     return
    1273     if save_path is None:
-> 1274         raise ValueError("Can't load save_path when it is None.")
    1275
    1276     if not checkpoint_management.checkpoint_exists(compat.as_text(save_path)):

ValueError: Can't load save_path when it is None.
```

A função `tf.train.latest_checkpoint(MODEL_PATH)` estava retornando `None`. Achei um problema parecido no StackOverflow no seguinte link:

<https://stackoverflow.com/questions/46652294/tensorflow-cannot-restore-model-couldnt-match-files-for-checkpoint>

Provavelmente estava com problemas para achar o checkpoint por causa do nome dos arquivos, que estavam como abaixo.



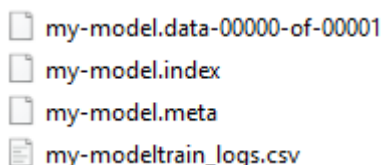
- .data-00000-of-00001
- .index
- .meta
- checkpoint

Além de um arquivo chamado `train_logs.csv`.

Por isso, no notebook de treino que já vimos anteriormente, fiz mais uma modificação. Coloquei um sufixo no `MODEL_PATH`.

```
1 MODEL_PATH = 'curatornet_10m/my-model'
```

Com isso, os arquivos ficaram da seguinte forma:



- my-model.data-00000-of-00001
- my-model.index
- my-model.meta
- my-modeltrain\_logs.csv

Então, no notebook de precomputação, coloquei o caminho absoluto do arquivo e ele não encontrou mais problemas.

```
1 MODEL_PATH = r'C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m'
2 MODEL_PATH

'C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m'
```

```
1 print(tf.train.latest_checkpoint(MODEL_PATH))
```

C:\Users\victor\_accete\CuratorNet\experiments\curatornet\_10m\my-model

Assim, o notebook de precomputação foi executado com sucesso.

```
In [69]: 1 item_vectors.shape
```

```
Out[69]: (13297, 200)
```

```
In [70]: 1 item_vectors.dump(MODEL_PATH + "item_vectors.npy")
2 with open(MODEL_PATH + 'ids', 'w') as f:
3     for _id in artwork_ids:
4         f.write('%s\n' % _id)
```

```
In [ ]: 1
```

## No notebook de Evaluation

Na última célula, coloquei o path completo para o arquivo de checkpoint.

```
In [*]: 1 run_experiments(artworks_dict, customers_dict, time_events,
2                       checkpoint_path=r'C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m',
3                       version=version,
4                       version_kwargs=dict(
5                           user_layer_units=[300,300,200],
6                           latent_space_dim=200,
7                           profile_pooling_mode='AVG+MAX',
8                       ))
```

E o notebook rodou sem erros.

## Metrics

O último notebook rodou sem erros de execução. A saída da última célula foi:

```
: 1 print_summary_table_regex('.*', ['auc_last', 'rec_at100'], 'auc_last', floatfmt='.6f')
name                                     auc_last    rec_at100
-----
._@10_(maxprofsz=oo)_curatornet_resnet_10m  0.543928    0.081516
```

No entanto, é um AUC abaixo do esperado. Tentei treinar o modelo mais algumas vezes e conferi que o arquivo gerado pelo notebook de evaluation continuou atualizando, mas mesmo assim o notebook de metrics continuou mostrando o mesmo resultado (com ligeiras variações).

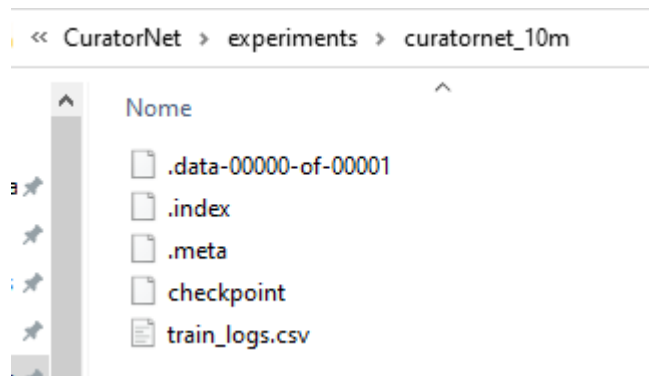
Acredito que talvez as mudanças que eu tive que fazer para rodar afetaram o código em fases posteriores.

## Novas tentativas

Atualizei mais uma vez o model\_path. Diferente do que estava no original e diferente da minha primeira modificação. Coloquei o caminho absoluto, mas sem o my-model, e com o indicativo de pasta no final.

```
: 1 MODEL_PATH = 'C:\\Users\\victor_accete\\CuratorNet\\experiments\\curatornet_10m\\'
```

Dessa vez, a estrutura de pastas ficou diferente.



Ele conseguiu ler corretamente o checkpoint. Ao rodar pela segunda vez, podemos ver que o modelo reagiu de uma maneira um pouco diferente dessa vez. Algumas épocas não geraram melhorias no modelo e com isso ele ia reduzindo a lr.

```
INFO:tensorflow:Restoring parameters from C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m\
model successfully restored from checkpoint!
Before training: test_accuracy = 0.936581
Starting training ...
EPOCH=1 => train_i=483, train_loss = 0.057671363670, test_accuracy = 0.9376679, epoch_secs = 69.34, total_secs = 69.34
** improvement detected: model saved to path C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m\
EPOCH=2 => train_i=483, train_loss = 0.091247674545, test_accuracy = 0.9298510, epoch_secs = 69.69, total_secs = 139.02
EPOCH=3 => train_i=483, train_loss = 0.115091852269, test_accuracy = 0.9215530, epoch_secs = 69.20, total_secs = 208.22
*** 2 checks with no improvements -> using a smaller learning_rate = 0.00006000
EPOCH=4 => train_i=483, train_loss = 0.125803321857, test_accuracy = 0.9301120, epoch_secs = 69.16, total_secs = 277.38
EPOCH=5 => train_i=483, train_loss = 0.124066190784, test_accuracy = 0.9476534, epoch_secs = 69.17, total_secs = 346.55
** improvement detected: model saved to path C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m\
EPOCH=6 => train_i=483, train_loss = 0.118495103287, test_accuracy = 0.9654907, epoch_secs = 69.22, total_secs = 415.77
** improvement detected: model saved to path C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m\
EPOCH=7 => train_i=483, train_loss = 0.112370313641, test_accuracy = 0.9654620, epoch_secs = 69.64, total_secs = 485.42
EPOCH=8 => train_i=483, train_loss = 0.107110547720, test_accuracy = 0.9605144, epoch_secs = 69.66, total_secs = 555.08
*** 2 checks with no improvements -> using a smaller learning_rate = 0.00003600
EPOCH=9 => train_i=483, train_loss = 0.102551220576, test_accuracy = 0.9648042, epoch_secs = 69.62, total_secs = 624.69
EPOCH=10 => train_i=483, train_loss = 0.098602924229, test_accuracy = 0.9542823, epoch_secs = 69.66, total_secs = 694.35
*** 2 checks with no improvements -> using a smaller learning_rate = 0.00002100
===== TIMEOUT =====
```

O notebook de precomputation, evaluation rodaram corretamente agora com o caminho absoluto e o indicativo de pasta no final do caminho.

```
INFO:tensorflow:Restoring parameters from C:\Users\victor_accete\CuratorNet\experiments\curatornet_10m\
----- starting experiment -----
500 tests done! elapsed time: 3.33 seconds
1000 tests done! elapsed time: 6.39 seconds
1500 tests done! elapsed time: 9.98 seconds
1978 tests done! elapsed time: 13.46 seconds
** recommendations successfully saved to ../results/@10_(maxprofsz=oo)_curatornet_resnet_10m.csv
```

No entanto, apesar de o resultado nas etapas anteriores terem sido animadores, mais uma vez, ao rodar o notebook metrics os resultados apresentados não são bons, contrariando os dados do treinamento e os dados do artigo.

1	print_summary_table_regex('.*', ['auc_last', 'rec_at100'], 'auc_last', floatfmt='.6f')		
name		auc_last	rec_at100
-----		-----	-----
._@10_(maxprofsz=oo)_curatornet_resnet_10m		0.547224	0.094841

Ainda não consegui identificar o motivo desse problema, mas abri uma issue no repositório do código do artigo. Até o momento não obtive resposta.

Pode ser que ainda haja alguma modificação que eu tive que fazer que gerou problemas em etapas subsequentes, ou pode ser um caso de overfitting também, que precisa ser testado.

# Tentativas de melhorias

Os autores responderam minha issue no Github e disseram que subiram uma atualização corrigindo um erro. No entanto, eles corrigiram o erro relacionado à parte de [Load Clusters](#), que eu já tinha superado.

Então, devido às diversas tentativas sem sucesso de conseguir obter bons resultados no notebook de métricas, foquei minhas atenções em tentar melhorar os resultados na etapa de treinamento.

Sei que devido a problemas de overfitting essa não é a melhor solução - mesmo usando um dataset de validação -, mas pode nos dar alguma ideia geral de quais medidas poderiam ser tomadas para melhorar o desempenho do modelo no dataset de testes.

## Baseline

Nosso baseline é o resultado inicial do notebook de treinamento. Aqui alguns dados sobre o modelo:

```
epochs=10,  
early_stopping_checks=2,  
weight_decay=.0001,  
learning_rates= [0.0001, 6e-05, 3.6e-05, 2.16e-05, 1.296e-05,  
7.776e-06, 4.6656e-06, 2.79936e-06, 1.679616e-06,  
1.0077696e-06]
```

E aqui os resultados que obtivemos ao fim das 10 épocas, que consideramos o nosso baseline:

```
train_loss = 0.222212419791,  
test_accuracy = 0.9205533,  
total_secs = 700.94
```

## Tentativa de melhoria 1: mais épocas

Devido ao código de treinamento ter um mecanismo de diminuir a learning rate, caso passe duas épocas sem melhoria na acurácia de teste, tentei triplicar o número de épocas. Isto é, aumentei de 10 para 30. Os demais hiperparâmetros permaneceram os mesmos, o resultado foi o seguinte:

```
train_loss = 0.078765004820,  
test_accuracy = 0.9866728,  
total_secs = 2176.15
```

Ou seja, com 30 épocas conseguimos resultados significativamente melhores. É importante notar que ao fim da trigésima época, o modelo detectou uma melhora. Isso significa que talvez o modelo apresentasse ainda mais melhoras, caso continuasse por mais épocas.

## Tentativa de melhoria 2: ligeiro aumento nas learning rates

A função provida pelos autores `get_decaying_learning_rates(1e-4, 1e-6, 0.6)` gerou as learning rates do que consideramos nosso baseline. Ajustei levemente os valores, para ter learning rates um pouco maiores, da seguinte forma: `get_decaying_learning_rates(1.1e-4, 1e-6, 0.62)`.

E obtive as seguintes learning rates:

```
[0.00011,  
 6.82e-05,  
 4.2284000000000004e-05,  
 2.6216080000000003e-05,  
 1.6253969600000002e-05,  
 1.0077461152000001e-05,  
 6.248025914240001e-06,  
 3.873776066828801e-06,  
 2.4017411614338566e-06,  
 1.489079520088991e-06]
```

Após isso, rodei o código de treinamento mais uma vez, sem carregar o checkpoint.

Com essa abordagem, na época 10 (mesma quantidade de épocas do baseline), obtivemos os seguintes resultados:

```
train_loss = 0.226793660833,  
test_accuracy = 0.9256272,  
total_secs = 728.49
```

Podemos observar que o tempo total foi um pouco maior (738.49s contra 700.94 da primeira abordagem), enquanto que houve um aumento de apenas aproximadamente 0.005 pontos percentuais de acurácia. Ou seja, não foi uma melhora significativa após a décima época.

Ao fim da da trigésima época, obtivemos os seguintes resultados:

```
train_loss = 0.078151677129,  
test_accuracy = 0.9854926,  
total_secs = 2178.34
```

Podemos observar que não há diferença significativa no tempo e a acurácia se tornou um pouco menor. Diante disso, não há motivos para acreditar, com as evidências disponíveis até o momento, que o ligeiro aumento na learning rate provoca uma melhora na acurácia.

## Tentativa de melhoria 3: muito mais épocas

Revertemos as learning rates para os mesmos valores do baseline. Para essa nova tentativa, tentamos aumentar o número de épocas para 100. Por motivos de recursos computacionais, optamos por não testar mais do que isso.

Obtivemos os seguintes resultados:

```
train_loss = 0.058502117424,  
test_accuracy = 0.9953534,  
total_secs = 7315.98
```

Podemos ver que houve uma melhora na acurácia, no entanto, isso pode ser devido a Overfitting, visto que os resultados do dataset de validação estão sendo usados para o treinamento.

Ao rodar um novo treinamento, rodamos por mais 20 épocas até que o modelo deu um early stopping, pois não houve melhora em nenhuma das épocas. A learning rate mais baixa foi usada (0.00000101), e mesmo assim não houve melhora. Por isso, aconteceu o early stopping.

## Tentativa de melhoria 4: learning rates menores

Dessa vez, usamos o modelo já pré-treinado por 100+20 épocas da tentativa de melhoria 3, mas com learning rates menores.

Usamos a função da seguinte forma: `get_decaying_learning_rates(1e-6, 1e-8, 0.6)` e obtivemos as seguintes learning rates:

```
[1e-06,  
6e-07,  
3.6e-07,  
2.1599999999999998e-07,  
1.2959999999999997e-07,  
7.775999999999998e-08,  
4.6655999999999984e-08,  
2.799359999999999e-08,  
1.6796159999999995e-08,  
1.0077695999999997e-08]
```

Recuperei o checkpoint do modelo treinado em 100 épocas com as novas learning rates menores e treinei por mais 50 épocas.

```
train_loss = 0.056190852655,  
test_accuracy = 0.9961511,  
total_secs = 3724.02
```

O tempo por época permaneceu praticamente o mesmo, mas dessa vez conseguimos uma pequena melhora na acurácia. Novamente, isso pode ser devido a Overfitting, visto que os resultados do dataset de validação estão sendo usados para o treinamento. Entretanto, essa estratégia de learning rates menores se mostrou promissora nesse quesito.

## Tentativa de melhoria 5: diminuindo o limiar de granularidade fina

Para esta abordagem, redefini o checkpoint mais uma vez para comparar com o baseline e com a tentativa de melhoria 1. `FINE_GRAINED_THRESHOLD` é uma variável que estava definida como **0.6** em todos os testes até agora. Mudamos o valor para **0.5** e treinamos por 30 épocas.

Ao chegar em 10 épocas, já podemos ver um problema com essa abordagem: o tempo de treinamento dobrou. Aqui estão os resultados com 10 épocas de treinamento:

```
train_loss = 0.129571539181,
```



```
test_accuracy = 0.9396193,  
total_secs = 1483.33
```

Podemos ver que a acurácia melhorou (93.96% comparado com 92.05% do baseline). É uma melhora interessante e pode ser explorada. Ao fim das 30 épocas, o resultado foi o seguinte:

```
train_loss = 0.058049695992,  
test_accuracy = 0.9937693,  
total_secs = 4489.12
```

A tentativa de melhoria 1 teve uma acurácia menor (98.67% contra 99.38% desta nova tentativa), mas em compensação o tempo mais do que dobrou. Ou seja, considerando o tempo de treinamento, talvez aumentar as épocas e manter o limiar de granularidade fosse igualmente efetivo.

## Tentativa de melhoria 6: aumentar o limiar de granularidade fina

Assim como diminuimos o limiar em 0.1 (reduzindo de 0.6 para 0.5), agora aumentamos de 0.6 para 0.7

Em 10 épocas, observamos os seguintes resultados:

```
train_loss = 0.095538463917,  
test_accuracy = 0.9349904,  
total_secs = 2243.93
```

Um pouco melhor que o baseline, mas com um tempo de treinamento muito maior (mais de 3x o tempo do baseline).

Ao fim da trigésima época, pudemos observar que o tempo de treinamento é muito alto, superando as 50 épocas da tentativa de melhoria 4.

```
train_loss = 0.053712114365,  
test_accuracy = 0.9962796,  
total_secs = 6656.01
```

Até o momento foi a melhor acurácia que atingimos. Portanto, pode ser que essa abordagem tenha a sua utilidade. O tempo de treinamento foi menor que a soma das tentativas 3 e 4 (a tentativa 4 usou o checkpoint da 3, por isso considero o tempo somado) e mesmo assim a acurácia foi um pouco melhor. Na minha avaliação, é uma abordagem interessante para se explorar.

## Tentativa de melhoria 7: aumentando a variável weight decay

Para essa abordagem, novamente redefini o checkpoint, para treinar do início. Aumentamos a variável `weight_decay` de 0.0001 para 0.0002 e treinamos o modelo.

Com essa nova configuração, o tempo de treinamento voltou a ser próximo do que vinha em todas as outras tentativas, exceto as tentativas 5 e 6.

```
train_loss = 0.305880842550,  
test_accuracy = 0.9117143,  
total_secs = 724.03
```

A acurácia teve uma ligeira diminuição em relação ao baseline. Ao fim das 30 épocas observamos os resultados:

```
train_loss = 0.129445350830,  
test_accuracy = 0.9863203,  
total_secs = 2187.96
```

Ou seja, não pudemos verificar mudança significativa no tempo de treinamento nem na acurácia em relação à tentativa de melhora 1.

## Conclusão

Após as análises feitas, acredito que as tentativas de melhora 4, 5 e 6 se mostraram as mais promissoras, baseado nos resultados obtidos. Infelizmente, o problema no notebook de métricas impossibilitou a avaliação em outras métricas com um dataset separado, e portanto fiquei dependente do notebook de treinamento. Apesar do treinamento ter um dataset separado para treinamento, o fato de estarmos vendo os resultados deste dataset enquanto mexia nos hiperparâmetros faz com que o modelo esteja sujeito a overfitting. Por essa razão, não considero aqui nenhuma abordagem como objetivamente melhor que as demais, apenas como uma abordagem promissora.