Treinamento de modelos de visão computacional

Abstract

Nestes documentos centralizadas as anotações e histórico detalhado do treinamento de alguns modelos de visão computacional.

- Proposta do projeto - Capstone Proposal
- Detalhes dos modelos desenvolvidos e exploração de dados
- Código Fonte

Organização

O trabalho foi dividido em vários notebooks para melhor organização.

Estes notebooks estão com a seguinte nomenclatura

```
<número sequencial técnica/modelo><refinamento>-<descricao>-
<base>
```

Ex:

- 01 - número sequencial
- b - refinamento
- Transfer Learning - técnica
- vazios - base de dados

Assim:

01-RedeSimples-chestXRay é uma rede neural simples para classificar a base chestXRay

01-RedeSimples-vazios é uma rede neural simples para classificar a base vazios

01b-RedeSimples-vazios é a mesma rede/técnica do 01 mas com algumas modificações

Modelos/bases

Conforme detalhado em CapstoneProject, serão treinadas redes convolucionais simples do zero, modelos sofisticados com transfer learning, e redes siameas. As bases utilizadas serão chestXRay, vazios e ncmsunicos.

== BASE ChestXRay =========

01-Baseline-redesimples-chestXRay

01-Baseline-redesimples-chestXRay

Rede convolucional bem simples treinada do zero.

Input shape = 150, 150

acc: 0.9279 - val_acc: 0.8285

01b-Baseline-redesimples-chestXRay-tamanhomaior

01b-Baseline-redesimples-chestXRay-tamanhomaior

Rede convolucional bem simples treinada do zero. Treinamento em 04/09/2019:

Foram realizadas várias rodadas(sempre continuando pesos do menor val_loss anterior):

- A primeira com ImageAugmentation e lr=0.001, melhor acc=0.94 e melhor val_acc=0.82 Mesmo a rede sendo simples, aparenta ligeiro overfitting
- A segunda com lr=0.0001 e mais épocas para os callbacks, melhor acc=0.94 e melhor val_acc=0.83
- A terceira sem ImageAugmentation, com lr muito pequena. Embora ImageAugmentation seja uma técnica para reduzir overfitting, e a priori tirar possa parecer contrasenso, apenas para
  testar se deixar a base de treinamento mais parecida com a de testes reduz erro de generalização, ao menos nesses exemplos e no "fine tunning"

Conforme previsto pela teoria, o sobreajuste aumentou.

acc foi para 0.96 e val_acc caiu para menos de 0.80

- Quarta tentativa, com regularização L1 e L2 na última camada e otimizador Adam, pareceu que ia conseguir melhoria, foi expandido o treinamento para 50 épocas iniciando com uma lr maior, mas a melhoria foi apenas marginal, com val_acc ensaiando ultrapassar 0.87 mas oscilando bastante

Em 04/06/2019 o melhor modelo foi:

Epoch 14/50 acc: 0.9507 val_acc: 0.8429

Conclusões/próximos passos

- Tentar aumentar regularização, utilizar keras-tuner
- Testar modelo pré-treinado mais poderoso (TransferLearning)
- Olhar exemplos de kernel no kaggle com melhor desempenho em busca de idéias

02c-TransferLearningSimples-FeatureExtractionRegularizer-chestXRay

02c-TransferLearningSimplesFeatureExtractionRegularizer-chestXRay

Utilizar DenseNet121 como feature extraction. Treinar classificador na saída desta rede.

Resultado testes: acc: 0.93 val_acc: 0.82

Próximo passo:

Gravar em .npy uma matriz com todas as features extraídas da base de treinamento e fazer Grid Search e Random Search do melhor classificador obtido.

02d-TransferLearning-FeatureExtraction-HyperParamTuner-chestXRay

02d-TransferLearningFeatureExtractionHyperParamTuner-chestXRay

Esta rede usa como entrada uma última camada maxpooling já salva, de saída da DenseNet121 aplicada à base de treinamento. Como todo o processamento convolucional já está realizado, o treinamento do classificador é centenas de vezes mais rápido. Assim, facilita o tunning da camada classificadora.

Resultado:

Foi possível obter um classificador utilizando somente a saída da DenseNet121 original com pesos da imagenet:

Base original: acc 0.95 val_acc 0.89 recall pneumonia: 0.95 0.97

02e-auxiliar-ImageAugmentation

02e-auxiliar-ImageAugmentation

Este notebook é apenas para gerar uma base aumentada pré-processada. Será utilizado pelo outro notebook 02e.

O objetivo é tentar diminuir o sobreajuste / distância entre acc e val_acc e agilizar a fase de treinamento.

02e-FineTunning-chestXRay

02e-FineTunning-chestXRay

Aqui está sendo treinada uma rede DenseNet121 do 02c empilhada com o classificador do 02d.

Problemas: não ficou claro se os pesos do notebook 02d foram aproveitados. Eles são carregados, os testes dão resultado similar ao 02d, mas quando inicia o treinamento de fine tunning os números de acc e val_acc caem próximos de

0.5, para depois voltarem a subir, mesmo quando se utiliza uma lr extremamente baixa.

Melhor modelo: Transfermodelweights02e_etapa2.02-0.66.hdf5

Base aumentada: acc 0.99 val_acc 0.83

Obs: Houve um problema, o acc na base train indica 99% no treinamento, mas estranhamente cai para 95% no relatório. Investigar.

Base original: acc 0.95 val_acc 0.89 recall pneumonia: 0.96 0.97

Observações finais

Considero que para este tipo de problema, o mais importante é um recall alto para pneumonia.

O modelo final tem um recall excelente, embora o desejável neste caso seja 100%, não sabemos se há erro de rotulagem nem qual o erro humano, muito menos o Bayes Error. Portanto, não dá para saber se é factível melhorar acima de 95-97% de recall.

Não foi possível obter ganhos significativos em relação ao baseline com as técnicas empregadas. A melhoria foi marginal, de menos de 5% em relação à rede neural simples. Tabela abaixo.

REDE 01b Accuracy: acc 0.95 val_acc 0.85 recall pneumonia: 0.94 0.95 REDE 02e Accuracy: acc 0.95 val_acc 0.89 recall pneumonia: 0.96 0.97

== BASE Vazios ==========

01-Baseline-redesimples-vazio

01-Baseline-redesimples-vazio

Rede convolucional bem simples treinada do zero.

acc: 0.9551 - val_acc: 0.9564

Este notebook também contém visualizações para tentar entender melhor o que foi aprendido pela rede.

01b-Baseline-redesimples-vazio-tamanhomaior

01b-Baseline-redesimples-vazio-tamanhomaior

Mesma rede convolucional, mas treinada com entrada maior (224x224). O tamanho de entrada é o mesmo da maioria dos modelos treinados na imagenet.

acc: 0.9589 - val_acc: 0.9616

Em 26/06/2019:

Rodada três vezes a sequência acima, 99, 101 e 103 erros de classificação (a mudança é devido a técnicas de image augmentation). Precisão de 100% na classe 0 e recall 91% ou seja 9% de erros tipo II falso negativo (predição 1 rótulo 0).

Analisando visualmente o diretório, pelo menos 25% dos erros são de rotulagem (os contêineres realmente não contém carga. Dos 70-75 erros restantes, em 20% do total o contêiner está escuro, parecendo ter carga de espuma. Em torno de 30% do total também há diversos tipos de ruídos na imagem, desde carretas que invadem a área do contêiner até borrões laterais na imagem, mas não carga. Então também é contêiner efetivamente vazio. Nos erros restantes (apenas 20% de 9%) parece haver erro de classificação, mas o contêiner contém pouca carga.

Conclusões:

```
* O erro real do algoritmo pode ser de apenas 2-4% e apenas na
classe Não Vazio.
Este erro poderia ser melhorado com melhora no recorte do
contêiner e na limpeza da imagem original.
* Dos 9% de erros, 2% são aparentemente "fraudes": contêineres
não continham carga
* Dos 9% de erros, 2% podem ser "fraude" ou falha no escâner
* Necessário proibir carretas que obstruam o contêiner
```

O algoritmo está tentendo a ignorar cargas de contêineres declarados como vazios mas borrados/sujos ou com muito pouca carga ou com carga uniforme

de espumas/materias pouco densos. Talvez fosse interessante forçar o algoritmo a ser mais tendente a diminuir este erro, mesmo que isto custasse aumento de falso positivo na classe vazio.

01b2-Baseline-redesimples-vazio-tamanhomaior-augmented-filtered

01b2-Baseline-redesimples-vazio-tamanhomaior-augmented-filtered

Este notebook aplica o mesmo método que 01b, mas trocando para base aumentada e filtrada (redução de erros de rótulo) produzida por 02c e o2d2, isto é, foi gerada nova base, já aumentada e excluindo erros acima e abaixo de um threshold do classificador 02c, que na inspeção visual ficou evidente tratarem-se de erros de rotulagem, isto é, data mismatch.

Base aumentada: acc: 0.97 - val_acc: 0.97 Base original: acc: 0.96 - val_acc: 0.96

01b3-Baseline-redesimples-vazio-tamanhomaior-augmented-filtered-menostranform

01b3-Baseline-redesimples-vazio-tamanhomaior-augmented-filtered-menostransform

Este notebook aplica o mesmo método que 01b, mas trocando para base aumentada e filtrada (redução de erros de rótulo) produzida por 02c e o2d2, isto é, foi gerada nova base, já aumentada e excluindo erros acima e abaixo de um threshold do classificador 02c, que na inspeção visual ficou evidente tratarem-se de erros de rotulagem, isto é, data mismatch.

Além disso, na inspeção visual do notebook 01b2 ficou a impressão de que os erros que ainda estavam ocorrendo eram: erros que mesmo o humano teria dificuldade (contêineres com espuma, por exemplo) ou erros de rótulo persistentes. Além desses, o algoritmo ainda erra em alguns poucos casos de contêiner contendo muito pouca carga, especialmente se esta se concentra apenas no solo (provavelmente confunde com imagens de vazio com solo poluído por carretas) ou somente em uma das portas (provavelmente

confundindo com reefer). Assim, neste notebook foi diminuída a amplitude das transformações de imagem aumentada para checar o resultado.

Base aumentada: acc: 0.97 - val_acc: 0.98 Base original: acc: 0.96 - val_acc: 0.96

02-TransferLearningSimples-vazio

02-TransferLearningSimples-vazio

Rede Densenet121, pré treinada na imagenet.

acc: 0.9545 - val_acc: 0.7126

Claramente, houve um sobreajuste muito grande. Os erros de classificação cometidos são gritantes.

Foi realizado fine tunning do último bloco convolucional (conv5):

acc: 0.9523 - val_acc: 0.8045

Apesar dos resultados ruins na generalização, necessário explorar mais esta possibilidade. A dificuldade pode ser devido ao bias em textura da imagenet. Note-se que esta base é em tons de cinza, e o mais importante é a geometria. Imagenet é colorida e textura é importante.

ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness https://arxiv.org/abs/1811.12231

02b-TransferLearningSimplesRegularizer-vazio

02b-TransferLearningSimplesRegularizer-vazio

Rede Densenet121, pré treinada na imagenet, com regularização.

02c-TransferLearning-FeatureExtractionRegularizer-vazio

02c-TransferLearningSimplesFeatureExtractionRegularizer-vazio

Rede Densenet121, pré treinada na imagenet, com regularização.

acc: 0.9408 - val_acc: 0.9514

Neste caso, se optou por utilizar as camadas pré treinadas para feature extraction, e, foi utilizada Max Pooling na última camada em vez de Avg Pooling.

Observações:

Após a extração das features das imagens, o treinamento do classificador é **centenas de vezes** mais rápido. Assim, a extração separada dos features permitirá treinar vários classificadores, fazer grid search e cross validation, entre outros.

Conforme demonstrado acima, há entre as imagens da classe nvazio diversos exemplos que parecem da classe vazio. Ou são erros de base ou são exemplos extremamente similares aos vazios. O aprendizado deve melhorar eliminando estes da base. Será criada uma cópia da base sem esses exemplos, para testar os mesmos algoritmos e comparar.

02c2-TransferLearningFeatureExtraction-Vazio

02c2-TransferLearningFeatureExtraction-Vazio

- Extrair features para numpy com imageaugmented bem "suave" (teste 01b3) produzida por 02c e o2d2
- Rodar com maxpool e com avgpool para poder comparar
- Rodar keras_tuner e comparar resultados com melhor resultado da rede simples

Base original maxpool: acc: 0.9604 - val_acc: 0.9566 Base original avgpool: acc: 0.9594 - val_acc: 0.9588

Parece que não importa o que se tente, há um platô em torno de 0.96 para accuracy na base original.

Com a base "limpa" de alguns erros de rotulagem, foi possível subir este platô para um pouco mais de 97%. Como a maioria dos erros é na classe vazio, antes de prosseguir: * Testar neste mesmo notebook treinamento com class_weigth * Copiar este notebook e repetir mesmos passos na base gerada por 02d2

O uso de class_weight 3 para a classe 0 (não vazio) causou queda marginal na accuracy total, mas distribuindo melhor os erros, conforme tabela abaixo (a accuracy caiu nas casas centesimais, em torno de 4 centésimos):

BASE TEST

Sem class_weight

```
          precision    recall  f1-score   support

     0.0       0.99      0.92      0.96      1166
     1.0       0.93      0.99      0.96      1138
```

Com class_weight

```
          precision    recall  f1-score   support

     0.0       0.97      0.94      0.95      1166
     1.0       0.94      0.97      0.95      1138
```

BASE TRAIN

Sem class_weight

```
          precision    recall  f1-score   support

     0.0       1.00      0.93      0.96     10494
     1.0       0.93      1.00      0.96     10306
```

Com class_weight

```
          precision    recall  f1-score   support
```

```
      0.0       0.98      0.95      0.96      10494
      1.0       0.95      0.98      0.96      10306
```

02c3-TransferLearningFeatureExtraction-Vazio

02c2-TransferLearningFeatureExtraction-Vazio

- Extrair features para numpy com imageaugmented bem "suave" e filtrado (mesma base que notebook 01b3)
- Rodar com maxpool e com avgpool para poder comparar
- Rodar keras_tuner e comparar resultados com melhor resultado da rede simples

Detalhes no notebook. Resumindo, os resultados foram muito similares ao notebook 01b3:

- aumento de 2% em accuracy em relação à base original, provavelmente pela correção de erros de rótulo
- De resto, resultados similares ao notebook 02c2, em todas as tabelas (com o aumento de quase 2%)

02d-auxiliar-ImageAugmentation-Vazios

02d-auxiliar-ImageAugmentation-Vazios

Notebook auxiliar para gerar uma base aumentada.

02d2-auxiliar-ImageAugmentationMenosTransfom-Vazios

02d2-auxiliar-ImageAugmentationMenosTransfom-Vazios

Notebook auxiliar para gerar uma base aumentada com poucas transformações.

02d2-auxiliar-ImageAugmentationMenosTransfom-Vazios

02d2-auxiliar-ImageAugmentationMenosTransfom-Vazios

Notebook auxiliar para gerar uma base aumentada com poucas transformações.

03-Busca-TransferLearning-Imagenet-Vazios.ipynb

03-Busca-TransferLearning-Imagenet-Vazios

Teste do uso das features extraídas de uma rede pré-treinada como hash para busca de similaridade.

Métricas utilizadas:

- Dos 10 primeiros e dos 10 20 primeiros resultados(de um total de 256 e 512 (, quantos pertencem à mesma classe?

Observações

Os resultados da rede simples treinada do zero foram similares ao uso de rede DenseNet, mas a extração de features com rede pré treinada na imagenet pode ser um método universal base para vários classificadores, buscas e análises.

Assim, quando uma imagem entrar no Banco de Dados, pré extrair as features via uma rede pré treinada, salvando no Banco de Dados, pode servir como ponto de entrada para vários tipos de classificadores e comparações, salvando memória e processamento posterior.

Os resultados utilizando maxpool e avgpool como extrator de características foram muito similares, com leve vantagem para avgpool nos resultados e menor tempo de convergência.

Desenvolvido na RFB dentro do escopo do Sistema AJNA

Ivan da Silva Brasílico

Código Fonte no GitHub

- Apresentado como Capstone Project no curso de Engenheiro de Machine Learning, Udacity.