Assignment 1: Transfer Learning

Juan Belieni e Juliana Carvalho

7 de setembro de 2023

Link para o notebook: Collab.

1 Dados

Neste assignment, trabalhamos com o conjunto de dados *beans* [1], que classifica imagens de folhas de feijão em três classes, duas que são relacionadas a doenças e uma para quando a folha é classificada como saudável.

Em todos os subconjuntos de dados (treinamento, validação e teste), as classes são aproximadamente igualmente representadas em quantidade. Ou seja, qualquer modelo minimamente viável deveria ter, ao menos, $\frac{1}{3}$ de acurácia, quantidade que usaremos posteriormente para considerar se cada modelo foi treinado corretamente e generalizou para o subconjunto de teste.

Antes do treinamento, o subconjunto de dados a ser utilizado no processo sofre um processo de *augmentation* [2], que aumenta a diversidade de dados e diminui as chances de o modelo sofrer *overfitting*.

Os *labels* de cada imagem são transformadas em um vetor de 3 dimensões por meio da função `tf.one_hot` [3], que codifica cada classe das imagens.

2 Modelo

Em cada um dos experimentos propostos, utilizamos a arquitetura convolucional da VGG16, adicionando mais uma MLP com 2 camadas:

- camada oculta densa de 128 nós e ativação ReLU;
- camada de saída densa de 3 nós e ativação SoftMax.

A função *categorical crossentropy* foi definida como a função de perda, por se tratar de um problema de classificação multi-classe.

3 Experimentos

Para todos os experimentos, utilizamos os seguintes hiper-parâmetros:

- 50 épocas;
- *batch size* igual a 32;
- taxa de aprendizado igual a 10^{-4}

Na tabela 1, é possível também visualizar outras características de cada experimento, como a inicialização dos pesos, as camadas congeladas e o número de parâmetros treináveis e não treináveis de cada modelo.

Exp.	Inicialização dos pesos	Camadas congeladas	Parâmetros treináveis	Parâmetros não treináveis
1	Aleatório	Nenhuma	14,780,739	0
2	*Imagenet*	Todas	66,051	14,714,688
3.a	*Imagenet*	Anteriores à `block5_conv1`	7,145,475	7,635,264
3.b	*Imagenet*	Anteriores à `block4_conv1`	13,045,251	1,735,488
3.c	*Imagenet*	Nenhuma	14,780,739	0

Tabela 1: detalhes do experimento.

4 Resultados

Os resultados de acurácia e perda para cada subconjunto de dados e tempo médio de época na fase de treinamento, presentes na tabela 2, mostram que o experimento contendo o melhor modelo foi o **3.a**. No entanto, todos os experimentos da etapa 3 obtiveram valores pareces parecidos de acurácia e perda e os experimentos **2** e **3.a** tiveram os menores tempos de treinamento.

	Treino			Validação		Teste	
Exp.	**Acurácia**	**Perda**	**Duração de época**	**Acurácia**	**Perda**	**Acurácia**	**Perda**
1	91.97%	0.208636	16s	90.98%	0.212813	89.84%	0.372857
2	92.07%	0.192317	7s	90.98%	0.223168	89.06%	0.283814
3.a	99.32%	0.018770	7s	95.49%	0.107799	96.88%	0.126274
3.b	97.87%	0.065777	9s	96.24%	0.144488	95.31%	0.245927
3.c	97.00%	0.109951	15s	93.23%	0.180557	96.09%	0.206707

Tabela 2: resultado dos experimentos, com os melhores valores de cada coluna sublinhados.

Pelos motivos citados acima, vamos investigar os resultados de cada experimento:

- **Experimento 1**: o modelo do experimento apresentou mais instabilidades nos valores de perda e na acurácia em relação aos outros experimentos (figura 1). A causa pode estar relacionada à inicialização aleatória dos pesos e também de um alto número de hiper-parâmetros (por conter todas as camadas convolucionais da VGG16 não congeladas).

  Ele também obteve a mais baixa acurácia em relação aos outros experimentos quando observamos a acurácia de treino. Isso demonstra que, nesse caso, utilizar os pesos do modelo VGG16 treinado na *imagenet* como base é preferível a uma inicialização aleatória dos pesos.

- **Experimento 2**: como é possível observar na imagem 2, esse modelo é o que apresenta a maior estabilidade da acurácia e da perda em relação aos outros modelos, o que pode ser devido à menor quantidade de parâmetros treináveis. No entanto, ele apresenta valores de acurácia e perda equivalentes ao primeiro experimento.

  Porém, o tempo médio de duração de época caiu pela metade. Ou seja, mesmo que o modelo não seja melhor que o anterior quando comparamos suas métricas, ele consome menos recursos no processo de treinamento, que acaba mostrando a eficácia da utilização de pesos do modelo

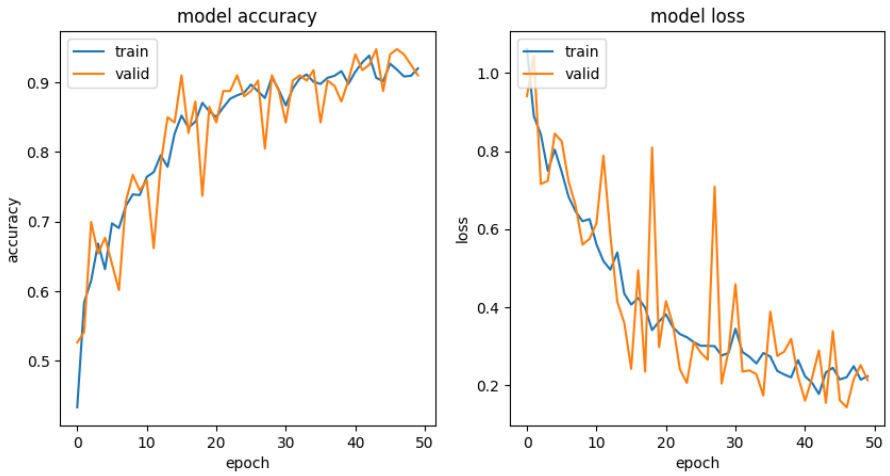

Figura 1: experimento 1.

treinado na *imagenet*. Veremos nos próximos experimentos como isso pode ser ajustado para tirar o melhor proveito dessa técnica.

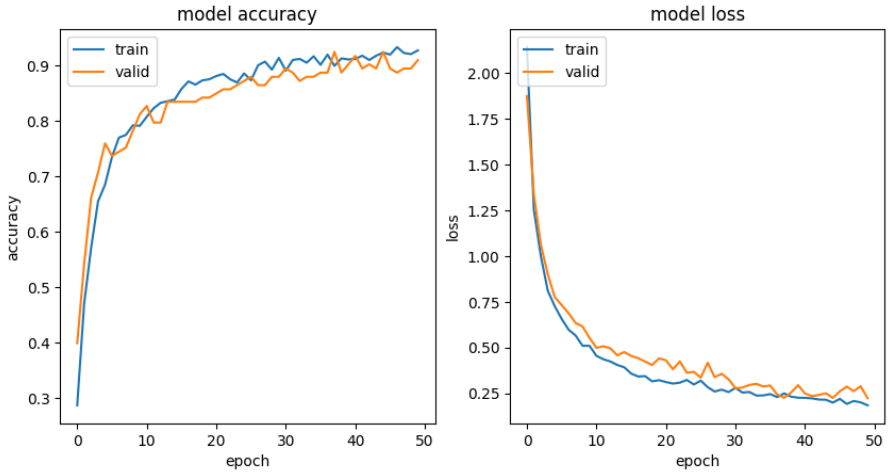

Figura 2: experimento 2.

- **Experimento 3.a**: esse foi o experimento que apresentou as menores perdas em todos os conjuntos de dados e a mais alta acurácia nos dados de treino e de teste. O bom desempenho pode estar relacionada ao fato do modelo apenas ter que aprender as últimas camadas convolucionais (além da MLP final).

  Como vimos em aula, as últimas camadas convolucionais normalmente são responsáveis por identificar os padrões mais complexos, enquanto as primeiras camadas identificam formas e objetos mais simples. Dessa forma, o modelo teve a oportunidade de ajustar essas camadas ao domínio do conjunto de dados com o qual estávamos trabalhando.

- **Experimento 3.b**: assim como no caso anterior, tivemos um bom resultado nas métricas, com destaque para a melhor acurácia nas imagens de validação. No entanto, começamos a observar um aumento no tempo e treinamento, que vai se repetir mais drasticamente no último experimento. Isso se deve à necessidade de ter que treinar mais pesos, se formos comparar com o experimento 2.

  No entanto, é possível essa maior quantidade de pesos disponíveis no treinamento beneficiasse o modelo com uma quantidade maior de épocas, pois ele poderia se ajustar melhor ao domínio

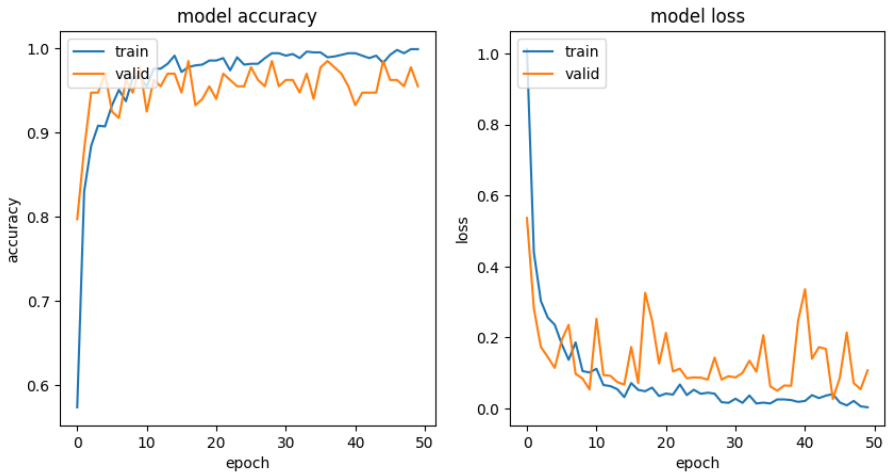

Figura 3: experimento 3.a.

dos dados.

- **Experimento 3.c**: tal qual os dois casos anteriores, apresentou um bom resultado nas métricas. Entretanto, a duração de época foi maior que os experimentos 3a e 3b e semelhante ao experimento 1, visto que a quantidade de parâmetros treináveis é igual a este primeiro experimento.

  Em relação ao experimento **1**, o único fator que mudamos foi a inicialização dos pesos com a Imagenet e a acurácia foi bem maior. Entretanto, o experimento desse modelo ainda foi inferior ao **3a** que teve camadas congeladas.

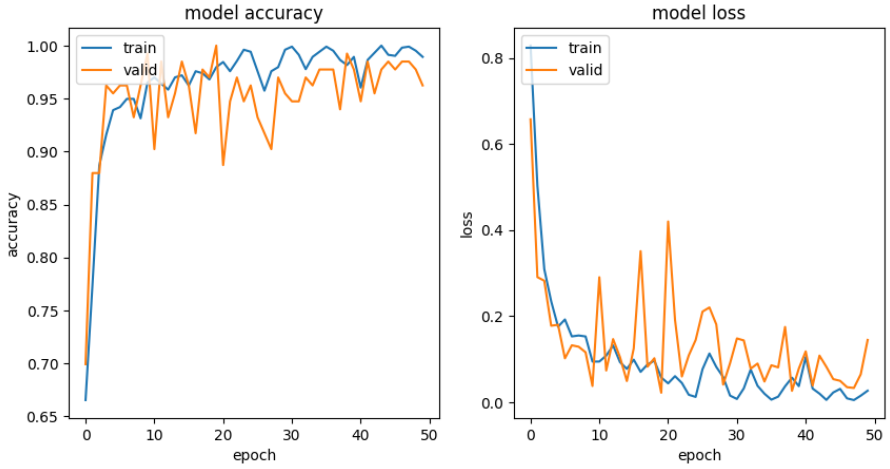

Figura 4: experimento 3.b.

- **Experimento 3.c**

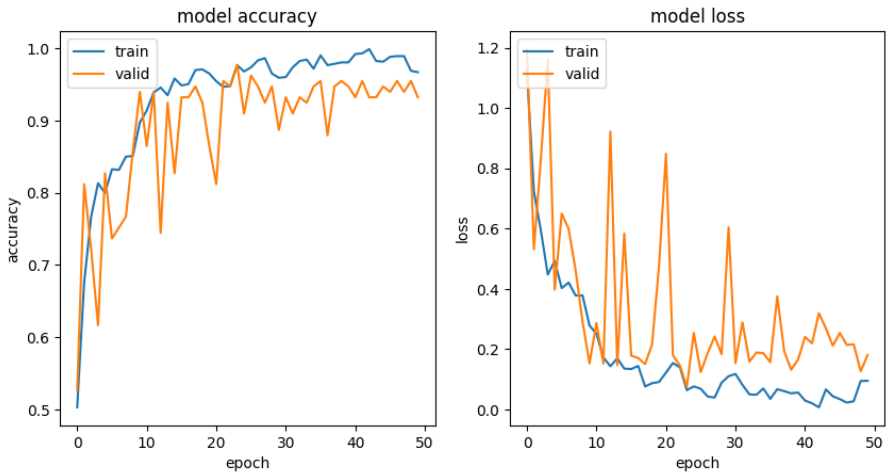

Figura 5: experimento 3.c.

5 Conclusão

Uma das principais conclusões que podemos extrair dos experimentos é que a utilização de pesos do treinamento em outra base de dados é capaz de oferecer um bom ponto de partida para um modelo de CNN.

No entanto, temos que levar em conta qual o nível de liberdade que o processo de treinamento terá para modificar esses pesos. Em um cenário com pouca liberdade (exemplificado no experimento **2**), o modelo pode não apresentar a melhor acurácia possível de ser alcançada e, com muita liberdade (exemplificado no experimento **3.c**), o modelo pode demorar um tempo desnecessário para ajustar todos os pesos.

Dessa forma, existe um ponto ótimo até onde devemos congelar os pesos de um modelo pré-treinado. Nos nossos experimentos, o modelo do experimento **3.c** foi o que obteve um dos melhores tempos de treinamento, além de alcançar os melhores valores nas métricas utilizadas para as análises.

Além disso, modelos com muitos pesos treináveis sofreram com muita instabilidade nas métricas ao longo do processo de treinamento. O experimento com a maior estabilidade desses valores foi o **2**, resultado que pode corroborar a primeira conclusão, já que essa baixa variância pode representar o limite do modelo de ajustar os pesos treináveis para encontrar uma configuração que tenha a melhor acurácia dentre os modelos.

Referências

[1] Makerere AI Lab. *Bean disease dataset*. Jan. de 2020. URL: https://github.com/AI-Lab-Makerere/ibean/.

[2] Tensorflow. *Data augmentation*. URL: https://www.tensorflow.org/tutorials/images/data_augmentation.

[3] Tensorflow. *tf.one_hot*. URL: https://www.tensorflow.org/api_docs/python/tf/one_hot.