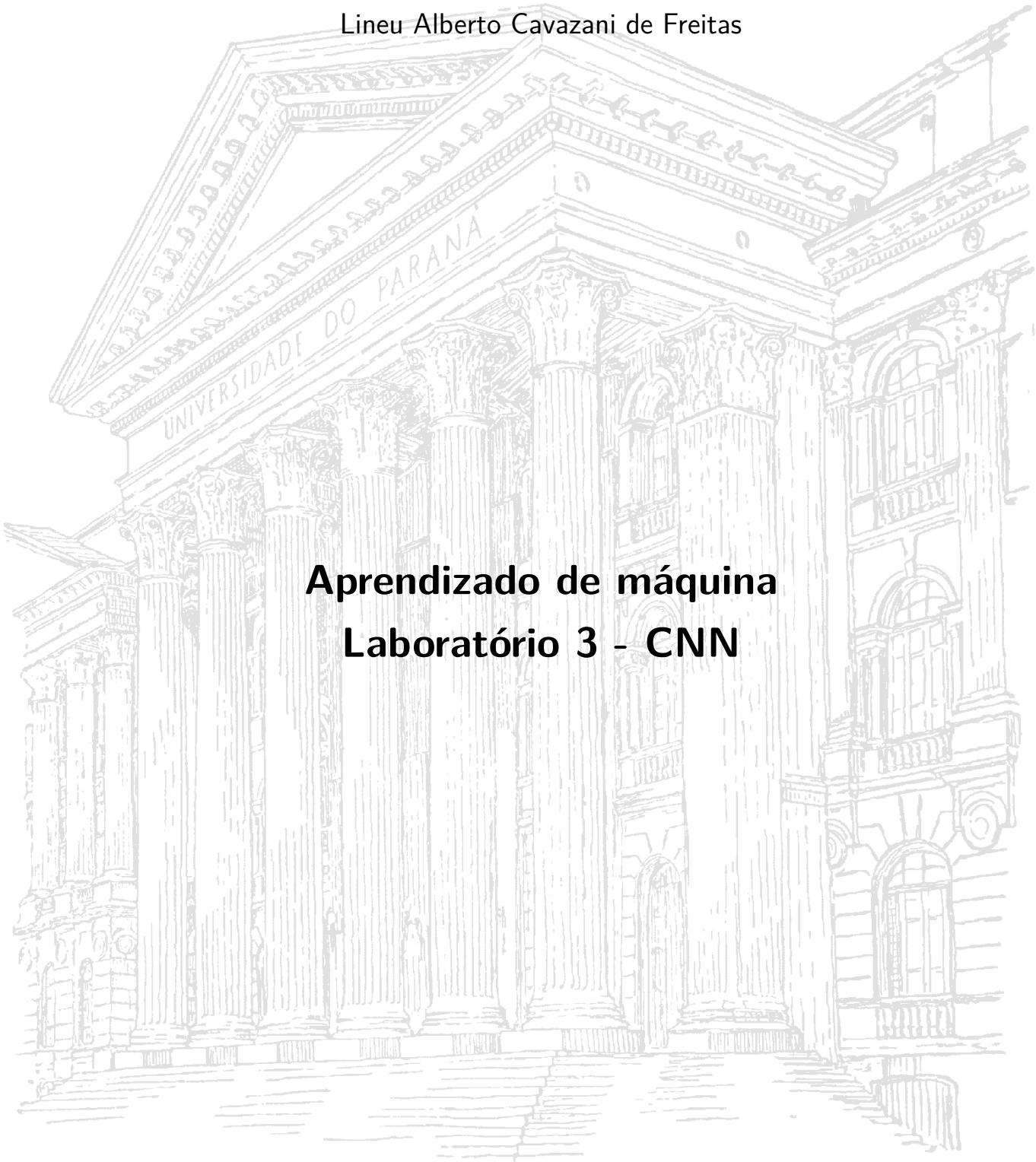
Universidade Federal do Paraná

Lineu Alberto Cavazani de Freitas



Aprendizado de máquina
Laboratório 3 - CNN

Curitiba

2021

Lineu Alberto Cavazani de Freitas

Aprendizado de máquina
Laboratório 3 - CNN

Relatório apresentado à disciplina Aprendizado de Máquina, ministrada pelo professor Luiz Eduardo Soares de Oliveira, no Programa de Pós Graduação em Informática da Universidade Federal do Paraná.

Universidade Federal do Paraná

Curitiba
2021

Sumário

1 Introdução

Redes Neurais Artificiais são técnicas de aprendizado de máquina formadas por modelos matemáticos que se inspiram na estrutura de um neurônio. Basicamente uma Rede Neural recebe uma entrada na primeira camada, esta entrada é associada a pesos, o resultado é passado para camadas subsequentes e por funções de ativação até que no final o modelo estime um valor predito do target para uma unidade com aquelas características.

Dentre as propostas de redes neurais, o Perceptron foi a primeira e mais primitiva estrutura. Trata-se de uma rede neural de um único neurônio capaz de lidar com problemas linearmente separáveis. Para problemas que não são linearmente separáveis, é possível aumentar a complexidade da rede através de camadas ocultas tal como ocorre no Multilayer Perceptron (MLP). Estas camadas extras podem ser vistas como extratores de características e, em geral, uma única camada escondida resolve a maior parte dos problemas. Um resultado interessante neste paradigma é que, dado uma quantidade suficiente de neurônios na camada escondida é possível resolver qualquer tipo de problema, desde que haja um bom vetor de características.

Uma desvantagem das Redes Neurais é que tratam-se de modelos superparametrizados, o que exige uma grande quantidade de exemplos de treino para resultados satisfatórios. Além disso, há a necessidade de definir o tamanho da rede, camadas, neurônios, taxa de aprendizagem, momentum, etc.

Uma outra utilidade de Redes neurais diz respeito à extração de características. A representação é, sem dúvidas, um grande dos gargalos em sistemas de aprendizado de máquina. Em geral, as características são definidas por um ser humano. Contudo, técnicas como representation learning ou feature learning vão no sentido de deixar a máquina definir a representação do objeto. Neste contexto, surgem as Redes Neurais Convolucionais (CNN), que servem tanto para predição como representação.

Para uma CNN usada para classificação de imagens, a imagem é a entrada da rede. A partir desta entrada extraem-se as características através de camadas de convolução. Nestas redes existem quatro tipos básicos de camadas: convolução, não linearidade (relu, função de ativação), pooling (agregação, redução do número de features) e classificação (fully connected layer).

O objetivo deste relatório é apresentar os resultados de um experimento que compara o desempenho de diferentes Redes Neurais Convolucionais a um problema de classificação de uma base de imagens de meses do ano manuscritos.

2 Descrição da atividade

A tarefa consiste em comparar o desempenho de diferentes redes neurais convolucionais conhecidas na literatura e também redes neurais pré treinadas para o problema de classificação de meses do ano manuscritos. Para o trabalho foi disponibilizado um conjunto já separado em treino e teste: a base de treino possuia 1578 imagens e a de teste 401.

Foram avaliadas duas redes neurais convolucionais clássicas: Lenet5 e AlexNet. Além destas duas redes foram testadas duas redes pré treinadas com pesos da Imagenet: Xception e MobileNetV2. Para comparação destas redes foi realizado um experimento no qual variou-se o batch size, número de épocas e learning rate das redes. As possibilidades testadas para cada uma destas variáveis foram:

- Batch size: 64, 128, 256.
- Épocas: 100, 300 e 500.
- Learning rate: 0.01, 0.1, 0.5.

O número de combinações únicas possíveis combinando cada nível é igual a 108. Cada combinação diz respeito a um modelo ajustado. Por se tratar de um problema balanceado a qualidade dos modelos foi aferida por meio da acurácia. Por meio de uma análise gráfica buscou-se verificar quais combinações apresentavam ajustes melhores ou piores. Após esta análise foram selecionados os modelos que apresentavam acurácia mais alta a fim de selecionar o modelo mais enxuto com bom desempenho. Para o modelo selecionado explorou-se a matriz de confusão e ainda utilizou-se este modelo para extração de características e aplicação desta representação a outro classificador, neste trabalho optou-se por uma regressão logística.

Os modelos foram ajustados utilizando a biblioteca scikit-learn, disponível para linguagem Python. Para execução do trabalho utilizou-se a plataforma Google Colaboratory (ou "Colab") que permite escrever código Python diretamente no navegador. Já a análise dos resultados foi feita com o software R. As parametrizações das redes foram feitas utilizando Keras e são descritas nas seções que seguem.

2.1 Lenet5

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
    input_shape=input_shape))
model.add(MaxPooling2D(strides=2))
model.add(Conv2D(filters=48, kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(strides=2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

2.2 AlexNet

```
model = Sequential()
model.add(Conv2D(filters=96, kernel_size=(11,11), strides=(4,4),
    activation='relu', input_shape=(227,227,3)))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(3,3), strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1),
    activation='relu', padding="same"))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(3,3), strides=(2,2)))
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
    activation='relu', padding="same"))
model.add(BatchNormalization())
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
    activation='relu', padding="same"))
model.add(BatchNormalization())
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
    activation='relu', padding="same"))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(3,3), strides=(2,2)))
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

2.3 Xception

```
xception_model = Xception(weights='imagenet', include_top=False)

## "Freeze" layers/weights
for layer in xception_model.layers[:]:
   layer.trainable = False

model = Sequential()
model.add(xception_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

2.4 MobileNetV2

```
mobile_model = MobileNetV2(weights='imagenet', include_top=False)

## "Freeze" layers/weights
for layer in mobile_model.layers[:]:
   layer.trainable = False

model = Sequential()
model.add(mobile_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

3 Resultados obtidos

As representações gráficas dos resultados dos experimentos são apresentadas na matriz de gráficos da Figura 1. As colunas da matriz representam as redes usadas e as linhas da matriz representam os learning rates usados. No eixo horizontal está representado o número de épocas de treinamento das redes. No eixo vertical é mostrada a acurácia. As cores representam cada batch size usado.

Os resultados mostram que não foram observadas acurácias muito distintas quando alterava-se o batch size, é possível verificar que, em vários momentos, os pontos ficaram sobrepostos.

Quanto ao learning rate, o único cenário que mostra algum efeito da taxa é aquele com valor igual 0.01 para a rede Lenet5; este cenário apresenta uma acurácia crescente conforme aumenta-se o número de épocas do treinamento. Nos demais cenários foi verificada uma considerável estabilidade dos resultados da acurácia, não sendo verificado uma melhora substancial no aumento de épocas do treinamento da rede.

Quanto aos tempos de ajuste, a rede Lenet5 se mostrou a mais rápida. A rede Alexnet levava cerca de 10 minutos por modelo. Para a rede pré treinada Xception, houve casos que demoararam mais de 30 minutos para execução dos cenários em 100 épocas, os cenários com treinamento maior que 100 épocas sequer puderam ser obtidos devido ao tempo. Já a rede pré treinada MobileNetV2 apresentou tempos similares ao Lenet5.

Quanto às acurácias em cada rede, os melhores resultados foram observados para a rede AlexNet, que apresentou acurácias acima de 0.9. A rede Lenet5 e o Xception apresentaram acurácias em torno de 0.8. Já a rede MobileNetV2 apresentou o pior resultado, com uma acurácia em torno de 0.4. A tabela 1 mostra as acurácias médias observadas para cada cenário.

Modelo	Lenet5	AlexNet	Xception	MobileNetV2
Média	0.79	0.93	0.77	0.40
Desvio padrão	0.05	0.02	0.007	0.005

Tabela 1 – Média e desvio padrão das acurácias de cada rede utilizada.

Com isso, a estratégia de seleção do melhor modelo para prosseguimento das análises se deu através do filtro dos modelos com acurácia acima de 0.9, com menor número de épocas, maior batch size e maior taxa de aprendizado. Este modelo foi um AlexNet com 100 épocas, learning rate de 0.5 e batch size de 256. A matriz de confusão deste modelo é apresentada na tabela 2.

	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**
Jan	**34**	4	0	0	0	1	0	0	0	0	0	0
Fev	1	**29**	0	0	0	0	0	0	1	1	0	0
Mar	0	0	**35**	0	1	0	0	0	0	0	0	0
Abr	0	0	0	**36**	1	1	0	1	0	0	0	0
Ma	0	1	2	2	**33**	0	0	0	0	0	0	0
Jun	0	1	0	0	0	**28**	0	0	0	0	0	0
Jul	0	0	0	0	1	2	**29**	0	0	0	0	0
Ago	0	1	1	0	0	0	0	**25**	0	0	1	0
Set	0	0	0	1	0	0	0	0	**27**	1	0	2
Out	0	0	0	0	0	0	0	0	0	**30**	0	0
Nov	0	0	0	0	0	0	0	0	0	0	**34**	0
Dez	0	1	0	0	0	0	0	0	0	0	6	**26**

Tabela 2 – Matriz de confusão do modelo AlexNet selecionado.

Este mesmo modelo foi ainda utilizado para extração de características e geração de novas bases de treino e teste para aplicação de um modelo de regressão logística para classificação dos meses manuscritos. Este modelo apresentou uma acurácia de 0.49 e a matriz de confusão é apresentada na tabela 3.

	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**
Jan	**16**	6	1	0	0	0	1	7	0	0	1	7
Fev	2	**14**	1	0	0	0	0	1	1	0	0	13
Mar	0	0	**35**	0	1	0	0	0	0	0	0	0
Abr	0	3	0	**14**	2	0	0	14	1	2	0	3
Ma	0	14	8	1	**2**	0	0	13	0	0	0	0
Jun	0	0	0	0	0	**0**	3	23	1	0	1	1
Jul	0	0	0	0	0	2	**23**	5	0	1	0	1
Ago	0	0	1	0	0	0	0	**27**	0	0	0	0
Set	0	0	0	0	0	0	0	1	**14**	2	0	14
Out	0	1	4	0	0	0	0	0	0	**22**	0	3
Nov	0	5	0	0	1	0	0	14	0	1	**3**	10
Dez	0	1	0	0	0	0	0	6	0	0	1	**25**

Tabela 3 – Matriz de confusão do modelo de regressão logística.

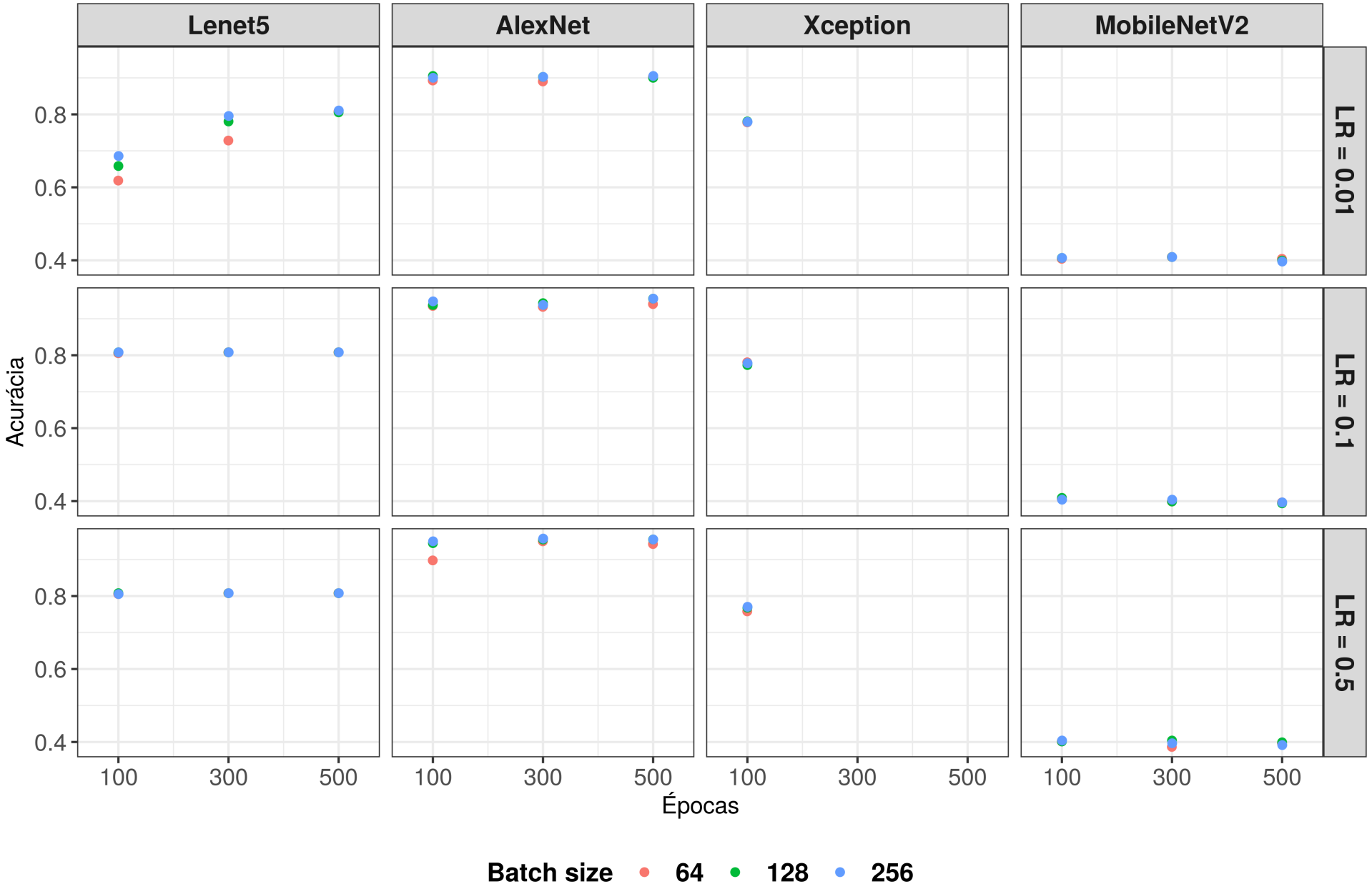


Figura 1 – Acurácia observada para cada combinação do experimento.

4 Considerações finais

Os resultados mostraram que a rede com melhor desempenho considerando acurácia como métrica de qualidade foi a AlexNet, com acurácias acima de 0.9. A segunda melhor rede foi a clássica Lenet5, com acurácias acima de 0.8.

As redes pré treinadas Xception e MobileNetV2 foram problemáticas. A rede Xception mostrou-se a mais demorada para ajuste e não apresentou desempenho superior às redes Lenet5 e AlexNet; também devido ao tempo não foi possível obter as acurácias para os cenários com mais de 100 épocas de treinamento. A rede MobileNetV2 apresentou ajuste rápido para todas as épocas, contudo os resultados para esta rede em termos de predição se mostraram extremamente pobres.

A dificuldade em utilizar o modelo Xception se dá pelo tamanho e complexidade da rede. Além disso, trata-se de uma rede pré-treinada para classificar imagens em 1000 categorias de objetos, o que pode justificar um resultado não muito satisfatório para classificação de dígitos manuscritos.

Já a MobileNetV2 é um modelo de classificação desenvolvido pelo Google que fornece recursos para classificação em tempo real em smartphones, o que pode justificar a velocidade do ajuste mas pobreza dos resultados neste contexto específico.

Quanto à utilização da melhor rede AlexNet para geração de um vetor de características para regressão logística, notou-se que o resultado da regressão logística foi consideravelmente pior do que aquele observado na classificação com a própria rede neural convolucional. Enquanto a AlexNet atingiu acurácia acima de 0.9, o modelo de regressão logística com entrada cedida pela rede apresentou acurácia de aproximadamente 0.5.

A principal restrição deste estudo é o tamanho da base de treinamento. Como estamos utilizando redes neurais convolucionais, sabe-se que existe a necessidade de uma grande quantidade de dados para trabalhar. A alternativa sugerida para este cenário é utilizar técnicas de Data Augmentation. Contudo, devido ao tempo demandado pelo experimento e ao tempo de disponibilidade das GPUs na plataforma utilizada, não foi possível explorar esta alternativa. Mesmo assim, ressalta-se um resultado satisfatório da rede AlexNet, com acurácia acima de 0.9 e da Lenet5, capaz de atingir acurácias máximas de 0.8 em modelos que demandam pouco tempo, devido à simplicidade da rede.