Computação para Análise de Dados

Equipe:
Renan Beserra
renan.beserra@ufrpe.br

Prof. Ermeson Andrade
ermeson.andrade@ufrpe.br

CNN
(Convolutional Neural Network)

Sumário

UFRPE
Universidade Federal Rural de Pernambuco

Introdução ao CNN

UFRPE
Universidade Federal Rural de Pernambuco

- As Redes Neurais Convolucionais (CNNs) são um tipo de arquitetura de rede neural projetada para processar dados com estrutura espacial, como imagens, sinais e séries temporais. Elas são amplamente utilizadas em tarefas como:
  - Classificação de imagens e sons;
  - Reconhecimento de objetos;
  - Detecção de anomalias;
  - Reconhecimento de padrões em sensores (IoT, dispositivos de borda).

Introdução ao CNN

UFRPE
Universidade
Federal Rural
de Pernambuco

- As CNNs extraem, hierarquicamente, informações espaciais através de camadas de convolução, pooling e, posteriormente, camadas totalmente conectadas para realizar tarefas de classificação ou regressão.
- Na classificação de imagens, por exemplo, as CNNs podem ajudar a identificar a qual classe a imagem pertence.

Vantagens e desvantagens

UFRPE
Universidade Federal Rural de Pernambuco

- Vantagens
  - Captura padrões espaciais.
  - Boa performance com muitos dados.
  - Não requer pré-processamento manual de características.
  - Excelente para tarefas de classificação e detecção em imagens.
  - Aprende desde características simples (bordas) até complexas (objetos).

Vantagens e desvantagens

UFRPE
Universidade Federal Rural de Pernambuco

- Desvantagens
  - Pode ser difícil de interpretar.
  - Necessita de muitos dados para treinamento eficiente.
  - Requer grande poder de processamento, especialmente para redes profundas.
  - Requer conhecimentos em ajuste de hiperparâmetros (como tamanho dos filtros, número de camadas, número de epochs) para se obter bom desempenho.

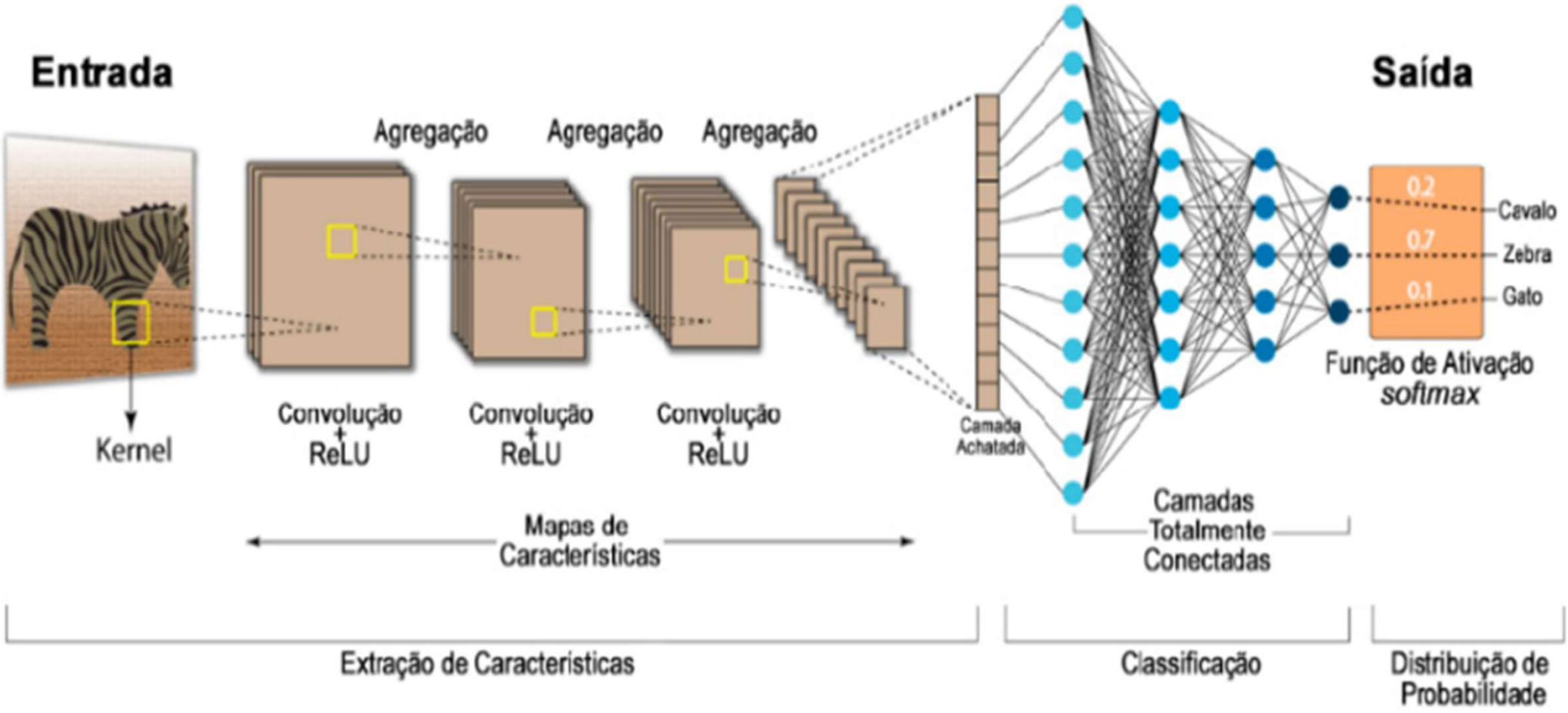
Etapas do CNN

UFRPE
Universidade Federal Rural de Pernambuco

1. **Entrada dos dados:** carregar e pré-processar sensores e imagens (redimensionar, normalizar).
2. **Camada Convolucional:** detectar padrões locais ou aplicar filtros para extrair características.
3. **Camada de Pooling:** reduzir a dimensionalidade e evita overfitting.
4. **Camada Fully Connected (densa):** aprender combinações complexas ou classificar as características extraídas.
5. **Saída:** obter a previsão final, classificação ou regressão (ex: classe da imagem).

Etapas do CNN

UFRPE
Universidade Federal Rural de Pernambuco



Sobre o dataset CIFAR10

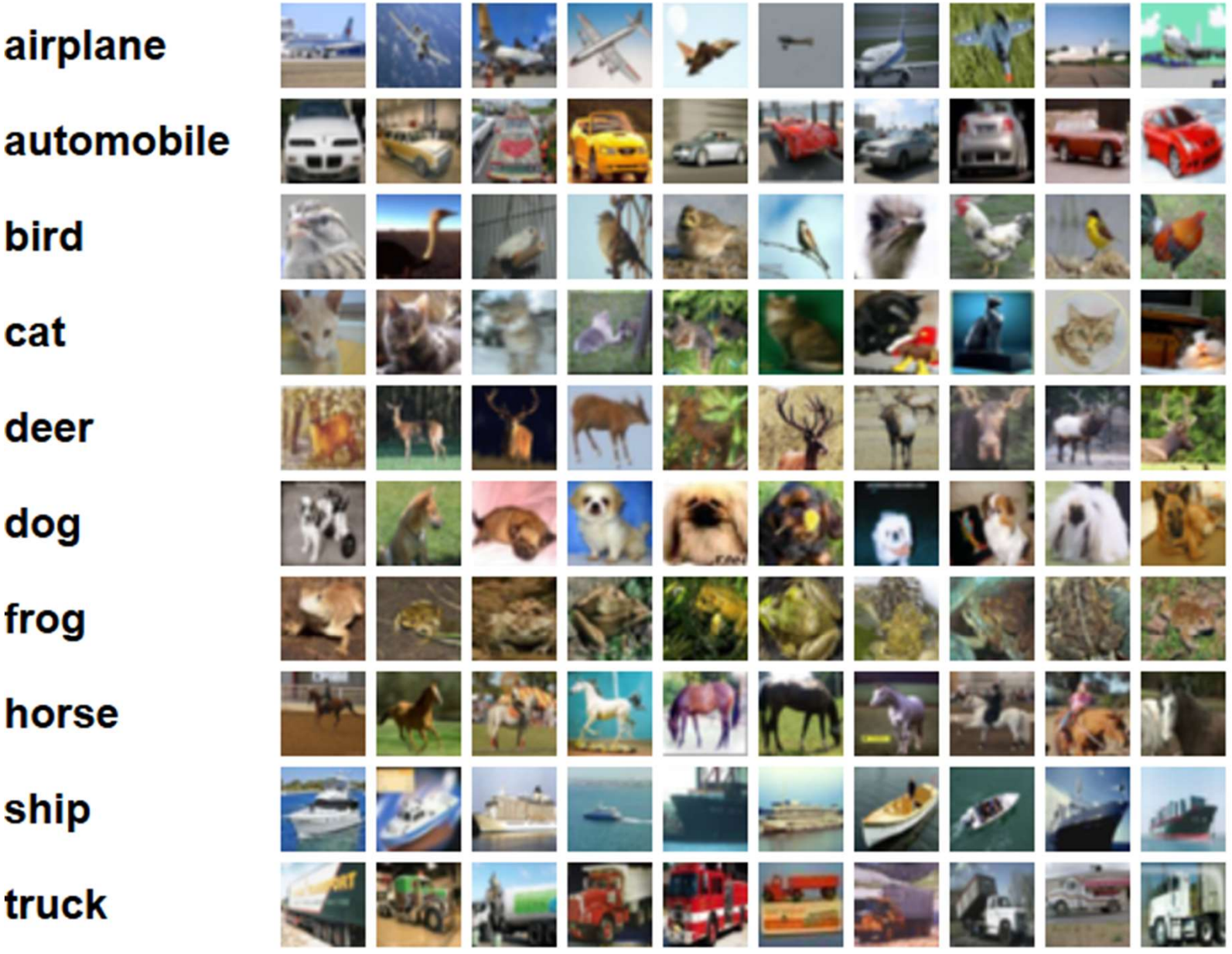
UFRPE
Universidade Federal Rural de Pernambuco

O CIFAR10 é um conjunto de dados clássico para visão computacional que contém:

- 60.000 imagens coloridas (32x32 pixels);
- Divididas em 10 classes (avião, automóvel, pássaro, gato, veado, cachorro, sapo, cavalo, navio, caminhão);
- 50.000 imagens para treino e 10.000 para teste;
- É um benchmark comum para testar algoritmos de classificação de imagens.

Sobre o dataset CIFAR10

UFRPE
Universidade Federal Rural de Pernambuco



Objetivo da implementação

Treinar uma rede neural convolucional (CNN) para classificar imagens do dataset CIFAR10 em suas 10 categorias, demonstrando:

- Carregar e preparar dados de imagem;
- Implementar uma arquitetura CNN;
- Treinar e avaliar o modelo;
- Aplicar a predição no modelo.

O que a implementação faz

UFRPE
Universidade Federal Rural de Pernambuco

- Instalação e Carregamento de Pacotes;
- Preparação dos Dados;
- Definição da Arquitetura CNN;
- Treinamento do Modelo;
- Avaliação dos Resultados;
- Carregamento e Pré-processamento de Imagens;
- Carregamento do Modelo Treinado;
- Carregamento e Predição da Imagem;
- Visualização de Imagens com Predição.

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Instalação dos pacotes e carregamento das bibliotetas

Entrada

```
# Instala os pacotes necessários
install.packages("torch")
install.packages("torchvision")
install.packages("luz")
install.packages("ggplot2")

# Carrega as bibliotecas
library(torch)
library(torchvision)
library(luz)
library(ggplot2)
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Carregamento e preparação dos dados

Entrada

```
# Define transformações para normalizar as imagens (usando médias e desvios padrão conhecidos do CIFAR10)

transform <- function(img) {
  img %>%
    transform_to_tensor() %>%
    transform_normalize(mean = c(0.4914, 0.4822, 0.4465),
                        std = c(0.2470, 0.2435, 0.2616))
}
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Carregamento e preparação dos dados

Entrada

```
# Baixa e carrega os conjuntos de treino


train_ds <- cifar10_dataset(
  root = "./data",
  train = TRUE,
  download = TRUE,
  transform = transform
)
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Carregamento e preparação dos dados

Entrada

```
# Baixa e carrega os conjuntos de teste


test_ds <- cifar10_dataset(
  root = "./data",
  train = FALSE,
  download = TRUE,
  transform = transform
)
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Carregamento e preparação dos dados

Entrada

```
# Cria dataloaders para alimentar os dados em batches
durante o treino


train_dl <- dataloader(train_ds, batch_size = 128,
shuffle = TRUE)
test_dl <- dataloader(test_ds, batch_size = 128)
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# Cria uma rede neural


net <- nn_module(
  "CIFAR10_CNN",
  initialize = function() {
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# 4 camadas convolucionais com ReLU


self$conv1 <- nn_conv2d(3, 32, kernel_size = 3, padding = 1)
self$conv2 <- nn_conv2d(32, 64, kernel_size = 3, padding = 1)
self$conv3 <- nn_conv2d(64, 128, kernel_size = 3, padding = 1)
self$conv4 <- nn_conv2d(128, 128, kernel_size = 3, padding = 1)
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# Camadas de max pooling e Dropout para regularização


self$pool <- nn_max_pool2d(2)
self$dropout <- nn_dropout(p = 0.5)
```

Implementação do CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# Camadas fully connected (calcularemos o tamanho
automaticamente)


self$fc1 <- NULL
self$fc2 <- nn_linear(512, 10)
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# Camada auxiliar para cálculo de dimensões
self$dim_calculator <- nn_sequential(
nn_conv2d(3, 32, kernel_size = 3, padding = 1),
nn_relu(),
nn_conv2d(32, 64, kernel_size = 3, padding = 1),
nn_relu(),
nn_max_pool2d(2),
nn_conv2d(64, 128, kernel_size = 3, padding = 1),
nn_relu(),
nn_conv2d(128, 128, kernel_size = 3, padding = 1),
nn_relu(),
nn_max_pool2d(2),
nn_flatten()    )  },
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
forward = function(x) {
# Calcular dimensões na primeira execução
if (is.null(self$fc1)) {
  test_output <- self$dim_calculator(x)
  input_size <- dim(test_output)[2]
  self$fc1 <- nn_linear(input_size, 512)$to(device = x$device)
}
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
x %>%
  # Bloco 1
  self$conv1() %>%
  nnf_relu() %>%
  self$conv2() %>%
  nnf_relu() %>%
  self$pool() %>%
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# Bloco 2
  self$conv3() %>%
  nnf_relu() %>%
  self$conv4() %>%
  nnf_relu() %>%
  self$pool() %>%
  self$dropout() %>%
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Definição da Arquitetura CNN

Entrada

```
# Achatar para a camada linear
torch_flatten(start_dim = 2) %>%


# Classificador
self$fc1() %>%
nnf_relu() %>%
self$dropout() %>%
self$fc2()
 }
)
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Treinamento do Modelo

Entrada

```
fitted <- net %>%

# Configura o modelo com a função de perda de entropia cruzada, com o otimizador Adam e com a métrica acurácia

  setup(

    loss = nn_cross_entropy_loss(),

    optimizer = optim_adam,

    metrics = list(luz_metric_accuracy())  ) %>%

  set_hparams() %>%
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Treinamento do Modelo

Entrada

```
# Treina por 20 épocas
  fit(
    train_dl,
    valid_data = test_dl,
    epochs = 20,
# Inclui callbacks para early stopping e checkpointing
    callbacks = list(
      luz_callback_early_stopping(patience = 3),
      luz_callback_model_checkpoint(path = "./models/")     )   )
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Treinamento do Modelo
  - Treinamento em progresso

```
Epoch 2/20
329/391 [==================================>------] - ETA:  1m - Loss: 1.4872  - Acc: 0.4491
```

Saída

```
Epoch 1/20
Train metrics: Loss: 1.8113 - Acc: 0.3201
Valid metrics: Loss: 1.4293 - Acc: 0.4776
...
Epoch 20/20
Train metrics: Loss: 0.764 - Acc: 0.7285
Valid metrics: Loss: 0.7109 - Acc: 0.7526
```

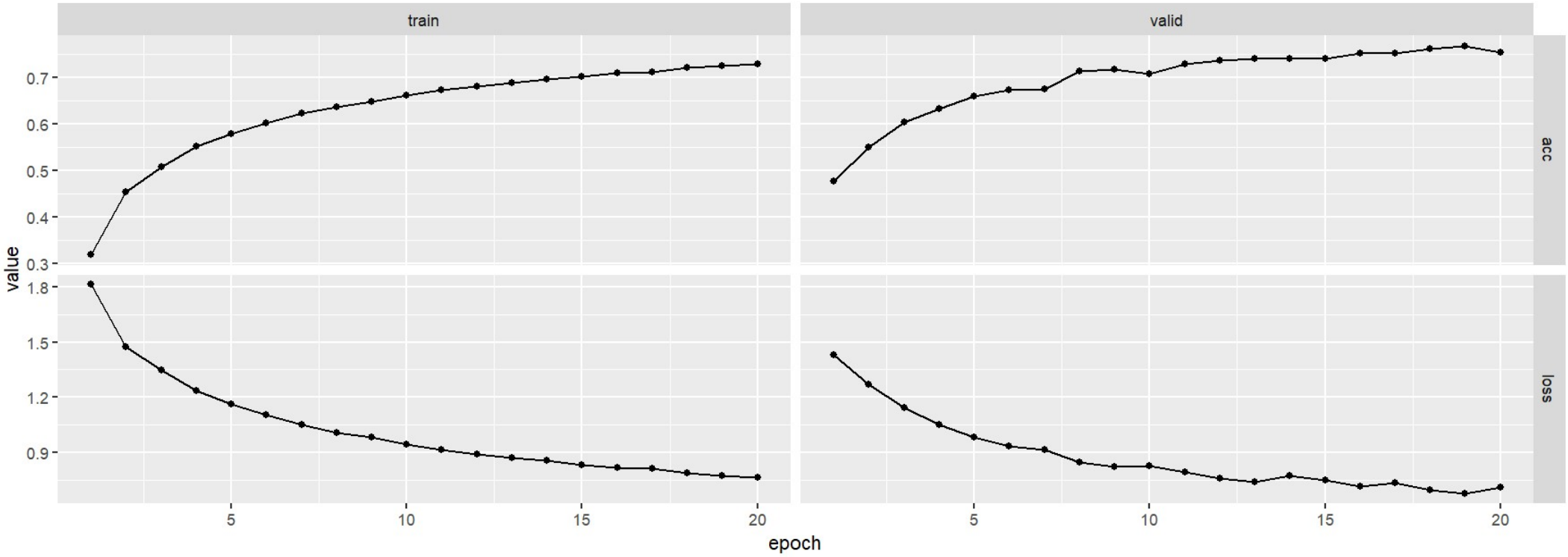
Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Avaliação dos Resultados

Entrada

```
# Mostra um gráfico com o progresso do treino
plot(fitted)
```



Implementação da CNN

- Avaliação dos Resultados

Entrada
`# Avalia o modelo no conjunto de teste` `evaluation <- fitted %>% evaluate(data = test_dl)`

- Progresso da avaliação

```
9/79 [===>--------------------------] - ETA:  1m - Loss: 0.7042  - Acc: 0.7526
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Avaliação dos Resultados

Entrada

```
# Mostra todas as métricas disponíveis
print(evaluation)
```

Saída

```
A `luz_module_evaluation`
--Results----------------------------------------------------------------
loss: 0.7109
acc: 0.7526
```

Implementação da CNN

UFRPE Universidade Federal Rural de Pernambuco

- Avaliação dos Resultados

Entrada

```
# Verifica a acurácia do teste
if (!is.null(evaluation$records$metrics$valid[[1]]$acc)) {
  test_acc <- evaluation$records$metrics$valid[[1]]$acc
  print(paste("Acurácia no teste:", round(test_acc * 100, 2), "%"))
} else {
  print("Não foi possível encontrar a métrica de acurácia")}
```

Saída

```
[1] "Acurácia no teste: 75.26 %"
```

Implementação da CNN

- Carregamento e Pré-processamento de Imagens

Entrada

```
# Função para pré-processamento
preprocess_image <- function(image_path) {
  # Carrega a imagem
  img <- image_read(image_path)
  # Redimensiona para 32x32 (tamanho do CIFAR10)
  img <- image_resize(img, "32x32!")
  # Converte para array numérico (0-255) e depois normaliza (0-1)
  img_array <- as.integer(img[[1]]) / 255
  # Reorganiza as dimensões para (C, H, W) - canais primeiro
  img_array <- aperm(img_array, c(3, 1, 2))
  # Converte para tensor torch
  img_tensor <- torch_tensor(img_array, dtype = torch_float32())
```

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Carregamento e Pré-processamento de Imagens

Entrada

```
  # Aplica normalização (usando os mesmos parâmetros do treino)
  transform_normalize(
    img_tensor,
    mean = c(0.4914, 0.4822, 0.4465),
    std = c(0.2470, 0.2435, 0.2616)
  )
}
```

Implementação da CNN

- Carregamento do Modelo Treinado

Entrada
`model <- fitted$model`

Implementação da CNN

UFRPE
Universidade Federal Rural de Pernambuco

- Carregamento e Predição da Imagem

Entrada

```
predict_image <- function(image_path) {
  # Classes do CIFAR10
  cifar10_classes <- c("avião", "automóvel", "pássaro", "gato”,
"veado", "cachorro", "sapo", "cavalo", "navio", "caminhão")
  # Pré-processa a imagem
  img_tensor <- preprocess_image(image_path)
  # Adiciona dimensão de batch (1, 3, 32, 32)
  img_tensor <-  img_tensor$unsqueeze(1)
```

Implementação da CNN

- Carregamento e Predição da Imagem

Entrada

```
  # Faz predição
  model$eval()
  with_no_grad({
  output <- model(img_tensor)
  probs <- nnf_softmax(output, dim = 2)
  pred <- torch_argmax(probs, dim = 2)  })
  # Retorna resultados
  list(
    class = cifar10_classes[as.integer(pred) + 1],
    probability = as.numeric(torch_max(probs)$item())
  )
}
```

Implementação da CNN

- Visualização de Imagem com Predição

Entrada

```
# Substitua pelo caminho da sua imagem
resultado <- predict_image("caminho da sua
imagem\\predicao_cavalo_CIFAR10.jpg")


cat(sprintf("Predição: %s (%.2f%% de confiança)\n", resultado$class,
resultado$probability * 100))
```

Saída

```
Predição: cavalo (99.97% de confiança)
```

Implementação da CNN

- Visualização de Imagem com Predição

Entrada

```
# Visualizar a imagem
image <- image_read("caminho da sua
imagem\\predicao_cavalo_CIFAR10.jpg")
print(image)
```

Exercício

O dataset MNIST é um grande banco de dados de dígitos manuscritos que é comumente usado para treinar vários sistemas de processamento de imagens. Ele tem um conjunto de treinamento de 60.000 imagens em tons de cinza 28x28 dos 10 dígitos e um conjunto de teste de 10.000 exemplos.

Exercício

UFRPE
Universidade Federal Rural de Pernambuco

- Configuração Inicial:
  - Instalar os pacotes se necessário:

```
install.packages("torch")
install.packages("torchvision")
install.packages("luz")
install.packages("ggplot2")
```

- Carregue as bibliotecas;
- Configure as sementes para reprodutibilidade;

Exercício

- Faça o pré-processamento dos dados:
  - Converta a imagem para tensor;
  - Normalize os valores com a média e o desvio padrão;
- Carregue o Dataset com mnist_dataset() para:
  - Conjunto de treino;
  - Conjunto de teste.
- Crie os DataLoaders com dataloader() para:
  - Conjunto de treino;
  - Conjunto de teste.

Exercício

- Defina a Arquitetura CNN com nn_module:
  - 3 camadas convolucionais com nn_conv2d();
  - 1 camada de Pooling com nn_max_pool2d() e 2 de Dropout com nn_dropout2d();
  - 2 camadas Fully Connected com nn_linear;
  - Fluxo de Dados (forward):
    - Primeiro bloco convolucional com nnf_relu();
    - Segundo bloco convolucional com nnf_relu();
    - Preparação para camadas densas com torch_flatten;
    - Camadas fully connected com nnf_relu() e nnf_log_softmax().

Exercício

UFRPE
Universidade Federal Rural de Pernambuco

- Configure o modelo com Luz (utilize o Otimizador Adam);
- Treine o modelo (determine 5 épocas);
- Mostre as métricas de treino e validação no gráfico;
- Avalie o conjunto de teste;
- Mostre a acurácia no teste;
- Carregamento e Pré-processamento de Imagens;
- Carregamento e Predição da Imagem;
- Visualização da Imagem com Predição.

Exercício

UFRPE
Universidade Federal Rural de Pernambuco

Sugestões de exercícios:

- Treine com uma época abaixo e outra acima do que você inseriu no exercício. Depois compare os resultados.
- Faça predições com novas imagens com fundo branco e número preto. Depois verifique se a classificação foi realizada corretamente.

Referências

- Bird J., Faria D., Manso L., Ayrosa P., &Ekárt A. (2021). A study on CNN image classification of EEG signals represented in 2D and 3D. Journal Of Neural Engineering, 18(2), 026005.
- Gupta J., Pathak S., Kumar G. (2022). Deep Learning (CNN) and Transfer Learning: A Review. Journal of Physics: Conference Series, 2273, 012029.
- C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.