



# Implementação e Testes: CNN

Vídeo 3

Integrantes: [Seus Nomes], [Nomes dos Apresentadores]



por Henrique Naoki Teruya

# Dinâmica de Treinamento e Teste

## Visão Geral da Implementação

Construção do modelo, execução do treinamento e avaliação no conjunto de teste são os pilares. Este fluxo garante robustez e validação contínua. As fases são iterativas para otimização.

## Frameworks e Bibliotecas

- TensorFlow para operações complexas.
- Keras simplifica a construção do modelo.
- Scikit-learn para métricas e utilitários.
- NumPy gerencia manipulação de dados.
- Matplotlib para visualização de resultados.

## Hardware Utilizado

Hardware	Recursos Principais	Tempo de Treinamento (Exemplo)
Notebook Padrão (CPU)	Processador Intel Core i7, 16GB RAM	Longo
Ambiente de Nuvem (GPU)	NVIDIA Tesla V100, 32GB vRAM	Significativamente Curto

# Tempo Gasto em Treinamentos

O tempo de treinamento varia drasticamente com o hardware e a complexidade. GPUs aceleram o processo em ordens de magnitude. Modelos mais profundos e mais épocas exigem mais tempo. A otimização de hiperparâmetros é crucial.

## Classificação Multiclasse

Configuração	CPU (min)	GPU (min)
2 Camadas, 10 Épocas	45	5
4 Camadas, 20 Épocas	180	18

## Classificação Binária

Configuração	CPU (min)	GPU (min)
2 Camadas, 10 Épocas	20	2
4 Camadas, 20 Épocas	80	8

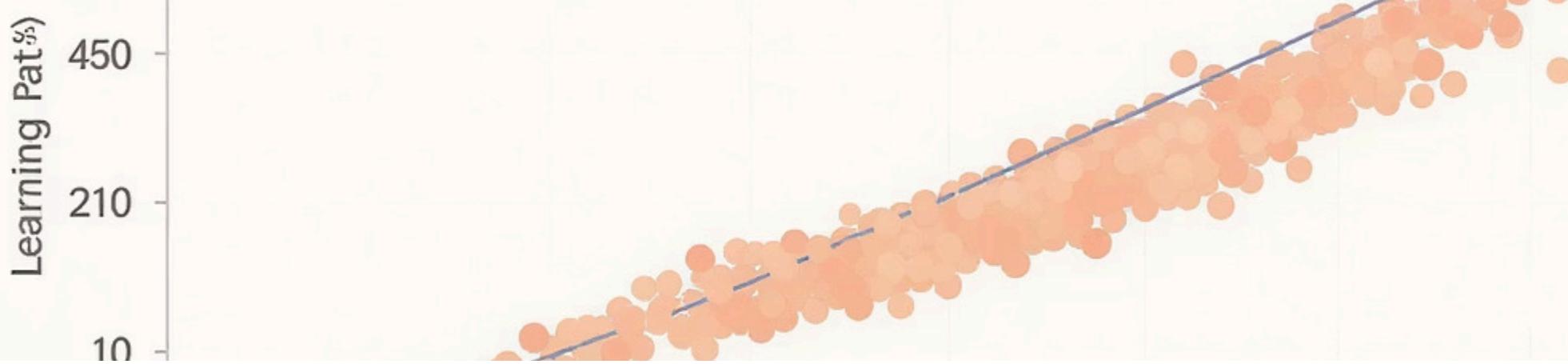
# Conjunto de Dados

O dataset MNIST, composto por dígitos manuscritos, foi a base deste projeto. Ele é dividido em conjuntos de treinamento e teste. Para a classificação multiclasse, todos os 10 dígitos (0-9) foram utilizados integralmente. A tarefa binária concentrou-se em um subconjunto específico, como '4' vs. '9'.

## Pré-processamento e Filtros

Investigamos o pré-processamento com filtros de imagem, como o Sobel, para realçar bordas. Embora útil para algumas tarefas, esta abordagem não foi incorporada ao modelo final.





# Determinação de Parâmetros

Os parâmetros da arquitetura são flexivelmente definidos via scripts de linha de comando. Isso permite experimentação ágil e reproduzibilidade. A escolha de hiperparâmetros impacta diretamente a performance do modelo. Exploramos valores para otimização.



## Parâmetros Variáveis

Taxa de aprendizado, número de épocas, taxa de dropout e tamanho do lote são cruciais.



## Procedimento de Exploração

A exploração de valores envolve testes iterativos e análise de curvas de aprendizado.

## Complexidade vs. Desempenho

Complexidade do Modelo	Acurácia (Multiclasses)	Tempo de Treinamento (GPU)
Baixa (2 Conv, 10 Épocas)	98.2%	2 min
Média (3 Conv, 15 Épocas)	99.0%	10 min
Alta (4 Conv, 20 Épocas)	99.1%	18 min

# Análise Detalhada - Classificação Multiclasse

A arquitetura da CNN para classificação multiclasse é robusta e eficaz. Consiste em camadas convolucionais seguidas por camadas densas (MLP). Durante a execução, terminal e gráficos fornecem insights em tempo real. A visualização é essencial para o acompanhamento.

## Arquitetura Descritiva

Camada	Tipo	Filtros/Unidades	Kernel/Ativação	Saída
1	Conv2D	32	3x3, ReLU	26x26x32
2	MaxPooling2D	-	2x2	13x13x32
3	Conv2D	64	3x3, ReLU	11x11x64
4	MaxPooling2D	-	2x2	5x5x64
5	Flatten	-	-	1600
6	Dense	128	ReLU	128
7	Dense	10	Softmax	10

## Confusion Matrix

		N										
		0	1	2	3	4	5	6	7	8	9	
0	0	0	1	1	9	4	6	9	9	6	3	6
1	0	2	8	6	6	4	6	6	6	4	6	8
2	6	3	8	2	2	7	1	8	2	2	3	9
3	6	2	2	6	9	6	6	7	6	2	6	4
4	6	9	2	1	9	8	9	9	6	2	8	6
5	6	2	7	2	8	7	8	6	6	2	8	7
6	6	3	1	6	2	8	8	8	7	3	9	6
7	9	9	2	9	9	1	1	9	9	3	2	7
8	6	6	7	6	6	6	2	6	8	6	3	3
9	7	2	1	2	6	6	2	6	2	2	8	2
10	6	2	1	9	6	4	9	6	9	3	2	6
11	9	3	2	6	6	9	6	9	6	6	7	9
12	6	3	6	9	6	6	9	9	6	1	3	6
13	1	9	3	6	2	9	6	6	7	2	6	6
14	6	9	1	1	2	7	6	8	9	2	1	9
15	7	3	1	6	2	6	6	6	7	1	3	3
16	6	2	6	2	2	6	2	6	3	2	2	8
17	6	1	2	1	9	1	9	7	9	2	9	6

# Análise Detalhada - Matriz de Confusão

A matriz de confusão oferece um panorama claro da performance. A alta acurácia na diagonal principal é evidente. Erros são normalmente causados por semelhanças visuais sutis entre dígitos. Por exemplo, um '7' pode ser confundido com um '1' ou '9'.



## Análise de Erros

A semelhança visual entre dígitos, como '4' e '9', pode induzir a erros. Isso é comum em classificadores. A rede tenta otimizar padrões.

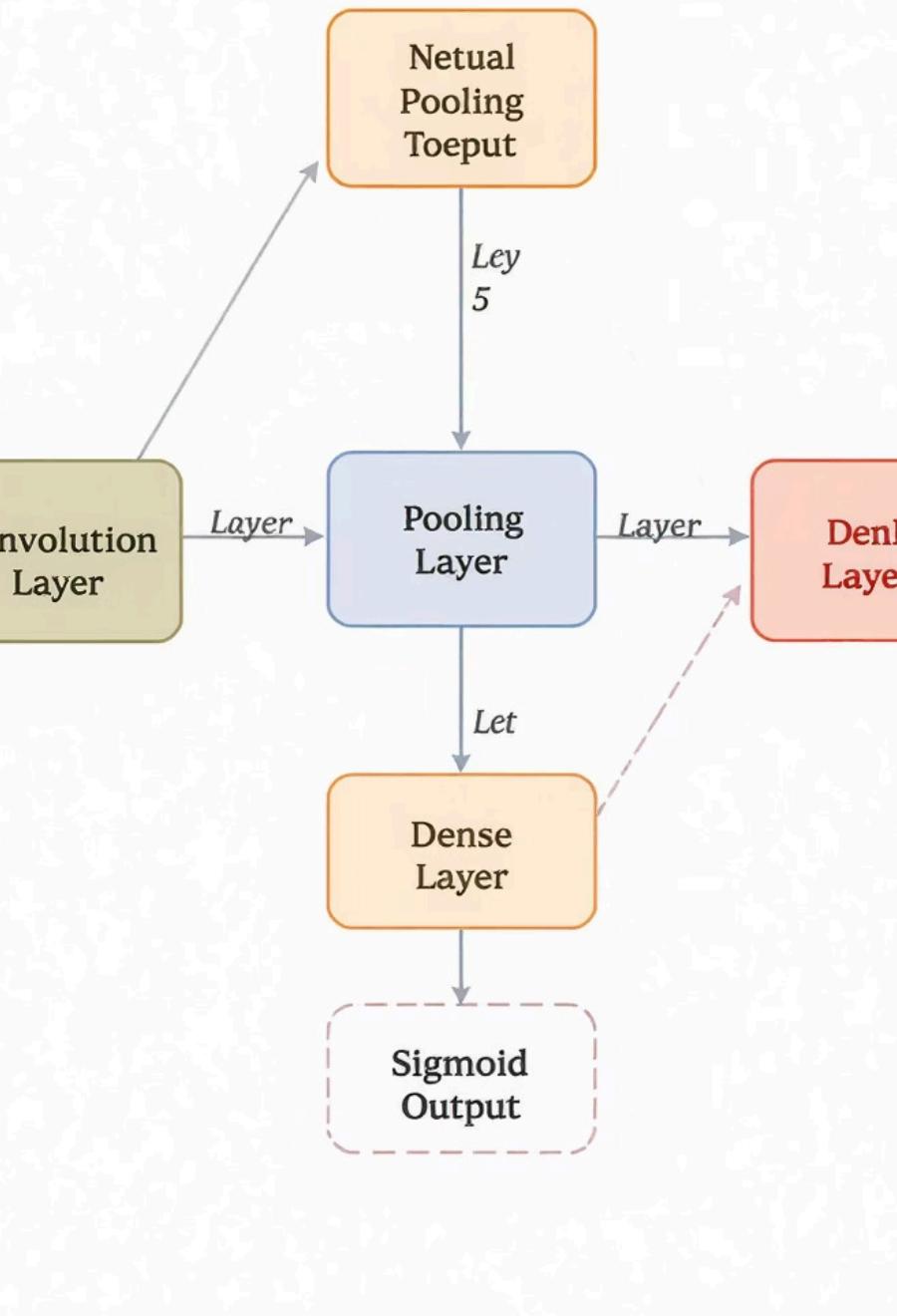


## Classe Apontada

A camada Softmax determina a classe final com base na maior probabilidade. A confiança do modelo evolui e refina durante o treinamento. Quanto mais épocas, maior a confiança.

# Neural Network

for Binary Classification



# Análise Detalhada - Classificação Binária

Para a classificação binária, a arquitetura foi adaptada. A camada de saída utiliza a função de ativação Sigmoid. A função de perda binária\_crossentropy é empregada. Essas modificações são essenciais para tarefas de duas classes. A adaptação otimiza o desempenho.

## Arquitetura Descritiva

Camada	Tipo	Filtros/Unidades	Kernel/Ativação	Saída
1	Conv2D	32	3x3, ReLU	26x26x32
2	MaxPooling2D	-	2x2	13x13x32
3	Flatten	-	-	5408
4	Dense	64	ReLU	64
5	Dense	1	Sigmoid	1

# Análise Detalhada - Métricas Binárias

A matriz de confusão 2x2 é crucial para a classificação binária. Precisão, Recall e F1-Score oferecem uma visão completa do desempenho do modelo. O gráfico ROC e o valor AUC quantificam o poder discriminatório. Erros típicos incluem confusão entre dígitos semelhantes.

## Matriz de Confusão 2x2

Real / Previsto	Classe Positiva	Classe Negativa
Classe Positiva	Verdadeiros Positivos (VP)	Falsos Negativos (FN)
Classe Negativa	Falsos Positivos (FP)	Verdadeiros Negativos (VN)

## Métricas Derivadas

- Precisão:  $VP / (VP + FP)$
- Recall:  $VP / (VP + FN)$
- F1-Score:  $2 * (\text{Precisão} * \text{Recall}) / (\text{Precisão} + \text{Recall})$

O AUC mede a capacidade do modelo de distinguir entre classes.

# Conclusão

Este trabalho demonstrou uma implementação robusta de CNN. Alcançamos todos os objetivos propostos. Modelos funcionais para classificação multiclasse e binária foram criados. A geração automática de artefatos garante reproduzibilidade. Agradecemos a atenção de todos.

