



UNIVERSIDADE FEDERAL DE SANTA CATARINA
GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO
SISTEMAS INTELIGENTES - INE5633

Lucas Broering dos Santos¹

ATIVIDADE PRÁTICA 2

Florianópolis, Setembro de 2025

¹ Graduando em Sistemas de Informação; Contato: lucas.broering@grad.ufsc.br

Sumário

Sumário.....	2
Itens.....	3
Tabela Comparativa e Análise dos resultados.....	7
Observações.....	8

Itens

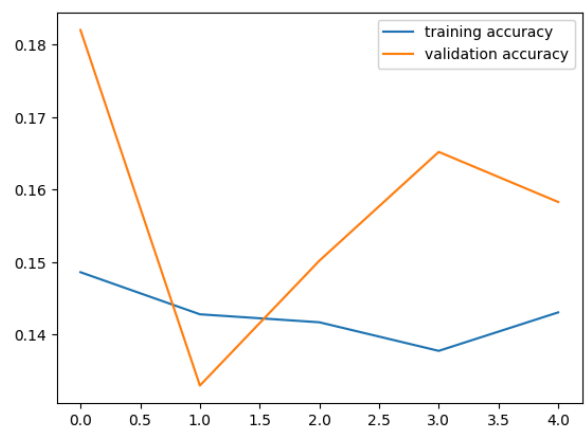
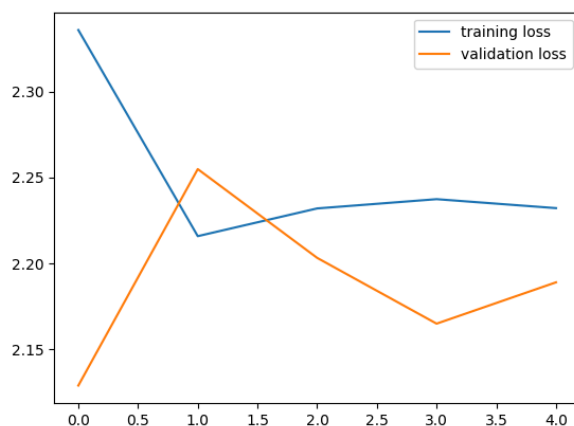
1 - Execute o treinamento da primeira rede neural e observe a relação entre as taxas de acurácia e de erro. Compreenda como são calculadas essas taxas e explique brevemente por que essas duas medidas não necessariamente (inclusive, dificilmente) são proporcionais.

A **acurácia** mede a frequência com que o modelo faz a previsão correta, sendo calculada pela razão entre o número de acertos e o total de amostras avaliadas. Enquanto o **erro** quantifica o quão distante as previsões estão dos valores reais, considerando também o **nível de confiança** do modelo, é calculado usando uma função como a cross-entropy. O **nível de confiança** representa a probabilidade atribuída pelo modelo à classe prevista. O erro leva esse valor em conta, penalizando mais previsões erradas feitas com **alta confiança** e menos previsões corretas feitas com **baixa confiança**.

Por esse motivo, acurácia e erro dificilmente são **proporcionais**: a acurácia apenas verifica se o modelo acertou ou errou, enquanto o erro avalia a **qualidade** da previsão, ou seja, o quão próxima ela está do valor real. Assim, é possível ter a mesma acurácia com erros muito diferentes, ou ter redução de erro sem aumento de acurácia, porque as duas métricas medem aspectos distintos.

2 - Interprete o erro e acurácia de treino e teste (especialmente a relação entre as medidas de treino e teste) de ambas redes neurais disponíveis. Essas medidas indicam que as redes estão boas? Há sobreajuste dos pesos? As redes já convergiram? Justifique sua resposta.

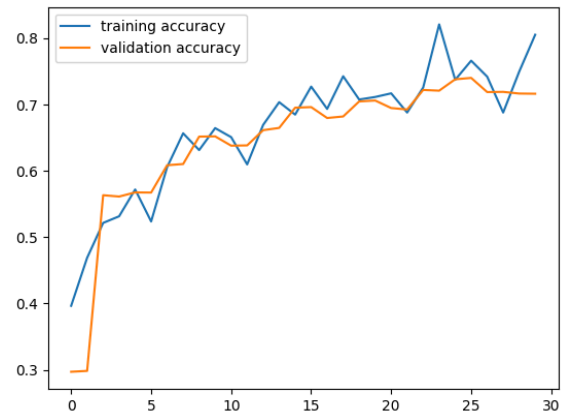
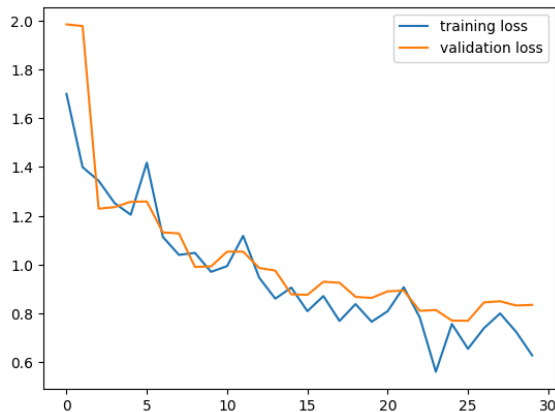
Primeiro vamos analisar os gráficos da **rede neural mais simples**, segue abaixo os gráficos de erro e acurácia:



A **primeira rede neural** apresentou acurácia muito baixa tanto no treino (entre 0,14 e 0,15) quanto na validação (entre 0,14 e 0,16), valores pouco superiores ao acaso (0,10). O erro de treino e validação permaneceu praticamente estacionado na faixa de 2,15 a 2,25, sem tendência clara de redução. Esse comportamento indica que a rede não conseguiu aprender padrões relevantes do CIFAR10.

Como as curvas de treino e validação são muito próximas, não há sinais de **sobreajuste**; o problema é de **subajuste**, pois a capacidade do modelo é **insuficiente** para a tarefa. Além disso, como nem a acurácia aumenta nem o erro diminui, concluímos que a rede **não convergiu**.

Agora vamos analisar a **rede neural convolucional profunda**, segue os dois gráficos abaixo:



A **segunda rede neural** apresenta uma evolução consistente tanto na acurácia quanto no erro. A acurácia de treino cresce até cerca de 0.80, enquanto a acurácia de validação estabiliza próxima de 0.73. As curvas permanecem próximas, indicando boa capacidade de generalização e **ausência de sobreajuste** significativo. Os gráficos de erro também mostram queda progressiva nas primeiras épocas, seguida de estabilização por volta da época 20, o que caracteriza a **convergência do modelo**. Portanto, essa rede apresenta desempenho adequado para o problema e evidencia que o uso de convolução e data augmentation melhorou substancialmente o aprendizado em relação ao primeiro modelo.

3 - A segunda rede neural, além de ter uma estrutura convolucional, utiliza aumento de dados. Qual a indicação do uso dessa técnica com relação à quantidade de dados por categoria?

O uso de aumento de dados indica que a quantidade de imagens por categoria não é **suficientemente grande para treinar a rede** de forma robusta, pois a pouca quantidade de dados aumenta o risco de **sobreajuste**: a rede pode memorizar as imagens de treino e não generalizar bem para exemplos novos. O aumento de dados atenua esse problema ao gerar, de **forma artificial**, novas imagens variadas a partir das originais, ampliando a diversidade do conjunto de treinamento. Essas variações podem incluir **rotações, espelhamento, zoom, translações, cortes aleatórios e ajustes de brilho ou contraste**, criando cenários diferentes que ajudam o modelo a aprender padrões mais robustos, em vez de depender de características específicas de cada imagem original.

4 - Para utilizar algum dos modelos treinados em produção, ou seja, receber uma imagem qualquer e classificá-la em alguma das categorias do CIFAR, quais etapas são necessárias?

Para utilizar um modelo treinado é necessário, **(1)** Importar o arquivo contendo a arquitetura e os pesos obtidos durante o treinamento, garantindo que a estrutura da rede seja restaurada corretamente, por exemplo um .h5. **(2)** Aplicar o mesmo pipeline usado no treino: redimensionar para 32×32 pixels, normalizar os valores dos pixels, ajustar o formato e garantir o mesmo canal de cor e escala numérica. **(3)** Enviar a imagem pré-processada ao modelo para obter o vetor de probabilidades correspondente às classes do CIFAR, usando `model.predict()` para obter o vetor de probabilidades para as 10 classes. **(4)** Interpretar a saída, identificando a classe mais provável e convertendo o índice previsto para o nome da categoria no CIFAR, **(5)** Por último, pode se dizer

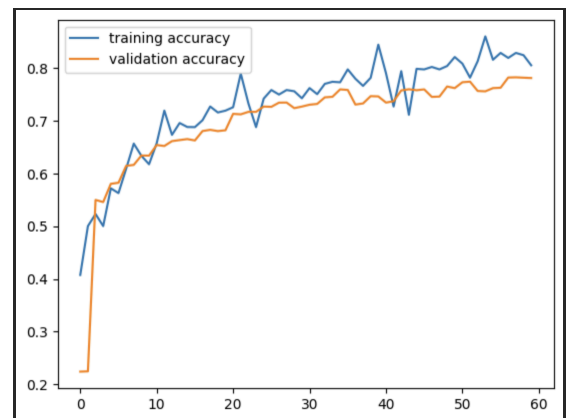
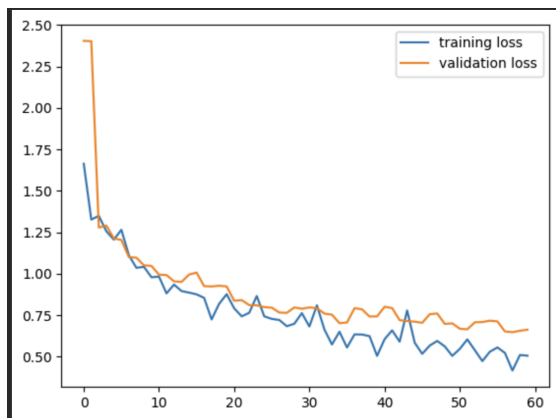
que é opcional, integrar em um serviço, por exemplo criar uma API com o modelo treinado, sendo de fácil acesso para um possível usuário.

5 - Os modelos disponibilizados são modelos simples de redes neurais, sendo que o primeiro está bastante limitado pois possui apenas uma camada oculta - e, conforme visto em aula, esta não é uma arquitetura adequada para classificar imagens. Escolha um dos modelos, preferencialmente o segundo, e:

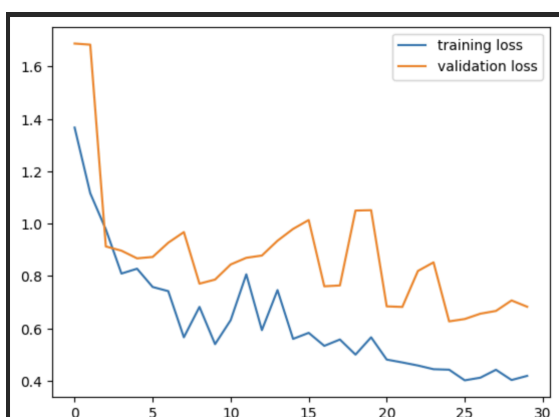
- Execute alguns testes na rede neural alterando sua estrutura (quantidade de camadas, quantidade de neurônios e diferentes funções de ativação).
- Teste diferentes opções para a fase de treinamento (especialmente taxa de aprendizagem e quantidade de épocas, porém a API disponibiliza mais parâmetros, seja criativo! - quanto mais nota precisar, mais criatividade e diferentes experimentos).
- Organize os testes realizados em uma tabela comparativa detalhando as combinações e os respectivos resultados.
- Analise os resultados explicando quais parâmetros / combinações mais interferiram na saída e qual o motivo.

Escolhi o segundo modelo para fazer algumas mudanças e testes, segue abaixo a tabela comparativa e os gráficos de erro e acurácia de cada modelo modificado. Os testes foram comparados com os gráficos de erro e acurácia do modelo padrão, tais gráficos estão em anexo na Questão (2).

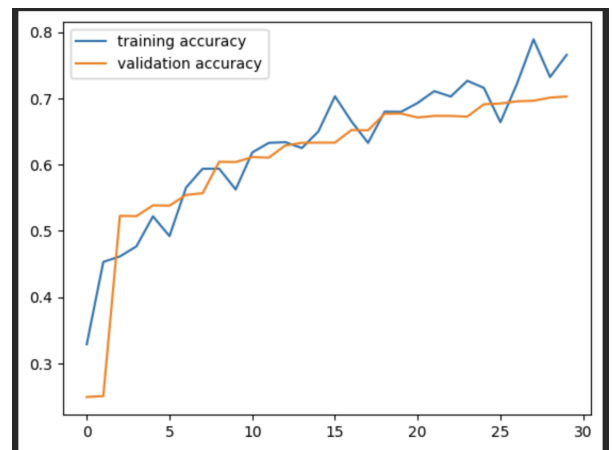
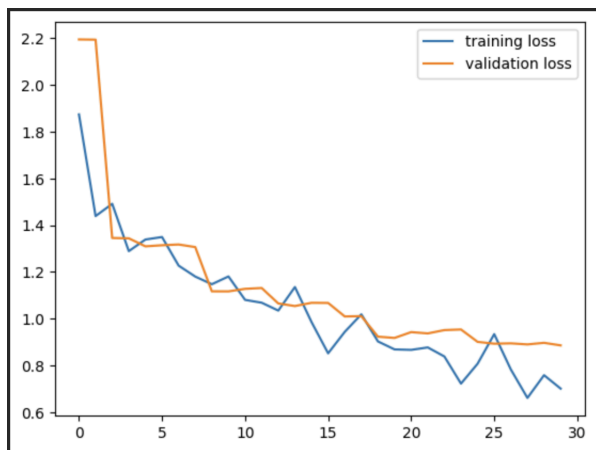
Teste 1 - Dobrar o número de epochs



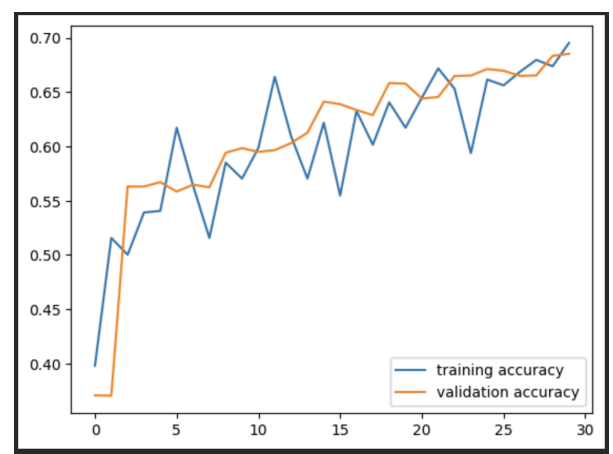
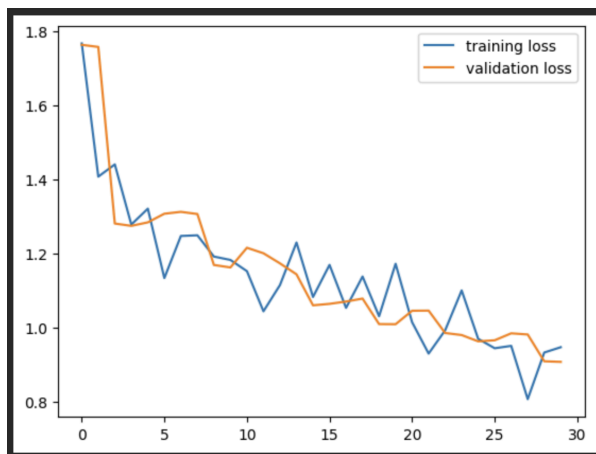
Teste 2 - Aumentar a taxa de aprendizagem para 0.001



Teste 3 - Adicionando mais 2 camadas convolucionais



Teste 4 - Ativação Selu



Teste 5 - SGD + momentum

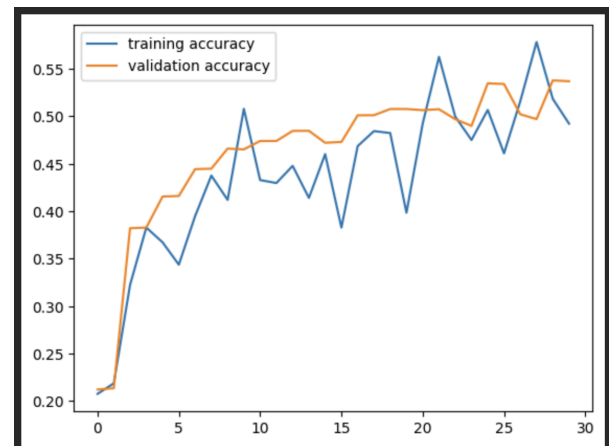
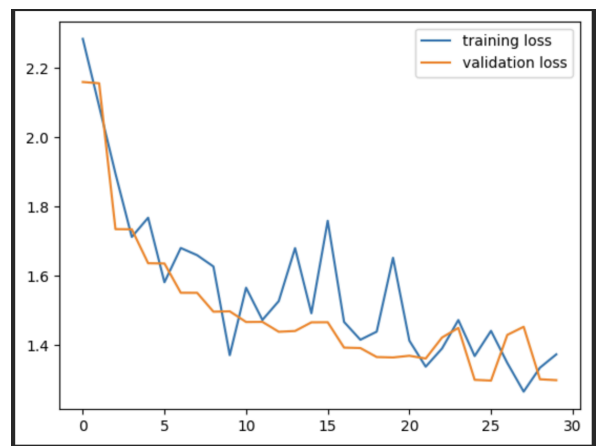


Tabela Comparativa e Análise dos resultados

Teste	Alteração	Acurácia	Erro	Observações
Teste 1	Dobrar o número de epochs	~72%	~0.75	Overfitting evidente após a época 40. Val loss sobe enquanto train loss continua caindo. A acurácia de validação estabiliza em ~72%.
Teste 2	Aumentar learning rate para 0.001	~60%	~1.0-1.2	LR muito alto causa instabilidade . Oscilações acentuadas em loss e acurácia.
Teste 3	Adicionar +2 camadas convolucionais	~70%	~0.7	Melhor estabilidade . Convergência suave, pouco overfitting. Arquitetura mais profunda aprende features mais eficientemente.
Teste 4	Trocar ativação para SELU	~62%	~0.8-0.9	Estabilidade no treinamento. Menor acurácia final mas melhor generalização. Gap mínimo entre treino/validação.
Teste 5	Otimizador SGD + momentum	~40%	~1.6-1.8	Pior desempenho . Convergência extremamente lenta. SGD não é adequado para esta arquitetura sem ajustes finos no LR.

Análise

Teste (1), **dobrar as épocas** mostrou que o modelo começou a sofrer **overfitting** após 40 épocas. A loss de validação parou de cair e começou a subir, enquanto a acurácia de treino continuou melhorando, indicando que o modelo estava memorizando os dados em vez de aprender padrões gerais.

Teste (2), **aumentar o learning rate para 0,001** causou instabilidade no treinamento. Tanto a loss quanto a acurácia ficaram **oscilando muito**, mostrando que a taxa estava alta demais e o modelo não conseguia **convergir adequadamente**, passando por cima dos pontos ótimos.

Teste (3), adicionar duas camadas convolucionais foi a mudança **mais benéfica**. A rede conseguiu aprender **features mais complexas e hierárquicas das imagens**, resultando numa **convergência estável** e na melhor combinação de acurácia e generalização entre todos os testes.

Teste (4), usar a **ativação SELU** trouxe muita estabilidade ao treinamento, com as curvas de treino e validação muito próximas, mas a acurácia final ficou mais baixa, sugerindo um trade-off entre **estabilidade e poder de representação**.

Teste (5), trocar para SGD com momentum foi a **pior alteração**. O treinamento ficou extremamente lento e a rede não conseguiu aprender padrões úteis, mostrando que esse otimizador precisa de um ajuste muito mais **cuidadoso dos hiperparâmetros para funcionar bem**.

Observações

Eu modifiquei o .ipynb, comentando qual parte do código está presente em cada teste, se usar um ctrl + f e escrever Teste, fica fácil de ver quais alterações fiz em cada teste. Segue o link do [.ipynb](#) modificado, também vou colocar o arquivo baixado no zip.