**Plano de ação**

**1) Objetivo e escopo**

* **Tarefa**: classificação de emoções em rostos (ex.: {raiva, nojo, medo, feliz, neutro, triste, surpresa}).
* **Entrada/Saída**: imagem 48×48 (ou 96×96) em tons de cinza → rótulo de emoção (softmax).
* **Métrica principal**: **macro-F1** (melhor que só acurácia quando a base é desbalanceada) + **matriz de confusão**.
* **Baseline**: um **SVM** ou **Logistic Regression** sobre *HOG* ou pixels achatados (scikit-learn). Isso dá um ponto de comparação “clássico” antes da CNN.

**2) Dados**

* **Dataset**: FER-2013 é ok para começar (já vem em 48×48 grayscale e classes de emoção).
* **Divisão**: treino/validação/teste (ex.: 80/10/10) — garanta **estratificação** por classe.
* **Normalização**: x = x/255.0 (faça também média/variância se usar modelos pré-treinados com 3 canais).
* **Aumento de dados** (data augmentation) moderado: leve *random crop/pad*, leve *rotation* (±10°), *brightness/contrast jitter* e *horizontal flip* (em geral é ok para emoções). Evite deformações fortes que alterem a semântica.

Dica: se optar por imagens RGB (3 canais) para *transfer learning*, apenas **repita o canal** (cinza→RGB) ou converta corretamente.

**3) Modelos**

1. **Baseline clássico** (scikit-learn) — simples e rápido.
2. **CNN do zero** (pequena, ~0.5–1.5M parâmetros) — conforme seu documento.
3. **Transfer learning** com uma backbone leve (ex.: **MobileNetV2**/**EfficientNet-B0**) — costuma ter melhor generalização com menos epochs.

**4) Arquitetura CNN**

* Blocos: Conv(32,3) → BN → ReLU → MaxPool × 2–3 blocos;  
  depois Conv(64/128,3) → BN → ReLU → GlobalAveragePooling → Dropout(0.3) → Dense(num\_classes, softmax).
* **Regularização**: Dropout, *weight decay* (L2=1e-4), *early stopping*.
* **Otimizador**: Adam (lr inicial 1e-3) com **redução de LR on plateau**.

**5) Treinamento e validação**

* **Batch**: 64 (ajuste conforme GPU/CPU).
* **Epochs**: treine com *early stopping* (paciência 7–10).
* **Callbacks**: *ModelCheckpoint* (melhor macro-F1 em validação) + *TensorBoard*.
* **Acompanhamento**: curva de perda e macro-F1; verifique *overfitting*.

**6) Avaliação**

* **Macro-F1** e **acurácia** no **conjunto de teste** (nunca use o teste para *tuning*).
* **Matriz de confusão** (quais emoções confundem mais).
* **Curvas**: loss/acc/F1 vs. epochs; tabela comparando *Baseline vs. CNN vs. Transfer*.
* **Ablations** rápidos\*\*: sem/ com augmentation; sem/ com BN; tamanho de imagem 48×48 vs 96×96.

**7) Ética e limitações**

* **Viés de dados**: distribuição de idade, gênero, tom de pele, iluminação; possíveis impactos.
* **Contexto**: expressão ≠ emoção interna garantida; ambiguidade cultural.
* **Privacidade**: uso apenas de bases públicas com termos claros; evite usar fotos pessoais.

**8) Reprodutibilidade**

* **Seeds fixas** (NumPy/TF/PyTorch).
* **Ambiente**: liste versões (Python, libs).
* **Script único** ou **Notebook** limpo com células numeradas e *requirements.txt*.

**Estrutura do relatório**

1. **Introdução**: problema, importância, objetivo, contribuições do seu trabalho.
2. **Trabalhos relacionados** (breve): FER-2013 e abordagens comuns (CNN/transfer).
3. **Metodologia**: dados, pré-processamento, divisões, modelos (baseline, CNN, TL), hiperparâmetros.
4. **Experimentos**: protocolo, *early stopping*, métricas, ablations.
5. **Resultados**: tabelas e figuras (macro-F1, acurácia, matriz de confusão; comparação entre modelos).
6. **Discussão**: análise crítica (erros típicos, confusões entre “medo”×“surpresa”, efeito do augmentation).
7. **Aspectos éticos/limitantes**: viés, privacidade, generalização.
8. **Conclusão e trabalhos futuros**: melhorias (detecção de face antes da classificação, atenção, ViT, *multi-task* com AUs, *mixup*, *label smoothing*).
9. **Reprodutibilidade**: link do repositório, versões, instruções para rodar.

**Checklist de entrega**

* **Código** limpo + requirements.txt.
* **README** com como rodar (treinar e avaliar).
* **Modelo salvo** (.keras/.h5 ou state\_dict.pt).
* **Relatório** com gráficos e tabelas.
* **Sementes** e versões fixadas.

**Riscos comuns (e como evitar)**

* **Overfitting forte** (treino ↑, validação ↓): use *early stopping*, Dropout, mais dados/augmentation.
* **Dados vazando** (data leakage): nunca faça *augmentation* no **teste**; não misture amostras entre splits.
* **Validação instável**: estratifique e repita com seeds diferentes (mostre média±desvio).
* **Classe rara ignorada**: foque em **macro-F1** e pese *class weights* se necessário.

**Glossário**

**Linguagens e Bibliotecas**

* **Python**: Linguagem de programação mais usada em IA, famosa pela simplicidade e pelas bibliotecas já prontas.
* **TensorFlow**: Biblioteca do Google para redes neurais, muito usada em aplicações reais (produção).
* **Keras**: Interface de alto nível (funciona em cima do TensorFlow). Facilita muito criar e treinar redes neurais.
* **PyTorch**: Biblioteca do Facebook (Meta), bastante popular em pesquisa e prototipagem rápida.
* **scikit-learn**: Biblioteca para aprendizado de máquina “clássico” (sem deep learning). Muito usada para pré-processamento e algoritmos básicos.

**Dados**

* **Dataset**: Conjunto de dados (imagens, textos, áudios etc.) usado para treinar e avaliar um modelo.
* **FER-2013**: Base de dados pública com imagens de rostos humanos e suas emoções rotuladas.
* **Pré-processamento**: Conjunto de passos para preparar os dados (redimensionar, normalizar, dividir em partes).
* **Normalização**: Ajustar valores de pixel (0–255) para uma escala padrão (ex.: 0 a 1).
* **Split (Treino/Validação/Teste)**:
  + **Treinamento**: onde a rede “aprende” (ajusta os pesos).
  + **Validação**: usado para calibrar o modelo e evitar *overfitting*.
  + **Teste**: avalia o desempenho final em dados novos.

**Arquitetura da Rede Neural**

* **Rede Neural Artificial (RNA)**: Modelo inspirado no cérebro, feito de “neurônios artificiais” conectados.
* **CNN (Convolutional Neural Network / Rede Neural Convolucional)**: Tipo de rede projetada para processar imagens.
* **Camada de Convolução (Conv2D)**: Extrai padrões da imagem (bordas, texturas).
* **Pooling (MaxPooling2D)**: Reduz a dimensão da imagem, mantendo as informações principais.
* **Camada Densa (Dense / Fully Connected)**: Cada neurônio conecta com todos os da camada anterior, usada para classificação final.
* **Softmax**: Função de ativação da camada de saída que transforma números em probabilidades (ex.: 70% feliz, 20% triste…).

**Treinamento**

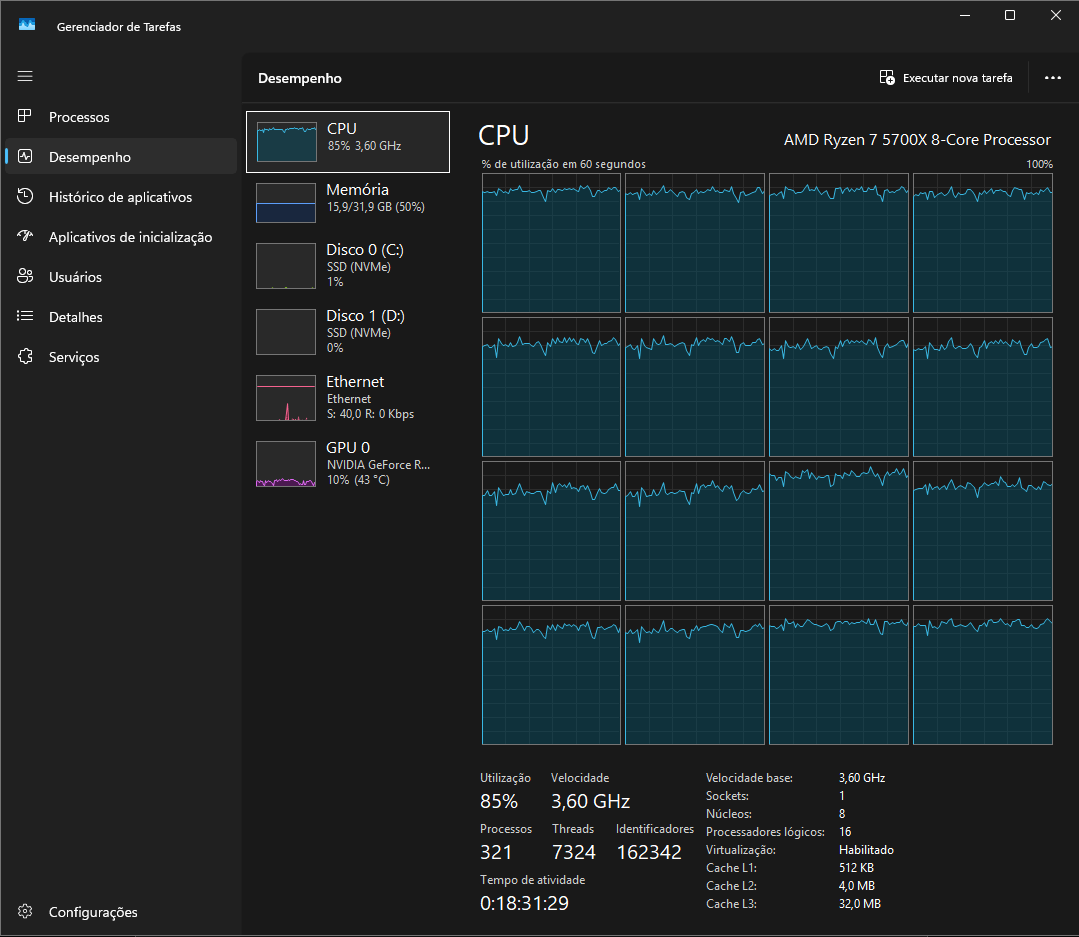
* **Época (Epoch)**: Uma passagem completa do dataset de treino pela rede.
* **Batch**: Subconjunto de exemplos processado de uma vez (em vez de carregar tudo de uma só vez).
* **Otimizador (Optimizer)**: Algoritmo que ajusta os pesos da rede (ex.: Adam).
* **Função de Perda (Loss Function)**: Mede o erro do modelo. Ex.: *categorical crossentropy* em classificação.
* **Métrica (Metric)**: Indicador usado para acompanhar o desempenho (ex.: acurácia, F1-score).
* **Overfitting (Sobreajuste)**: Quando a rede aprende muito bem os dados de treino, mas vai mal em dados novos.

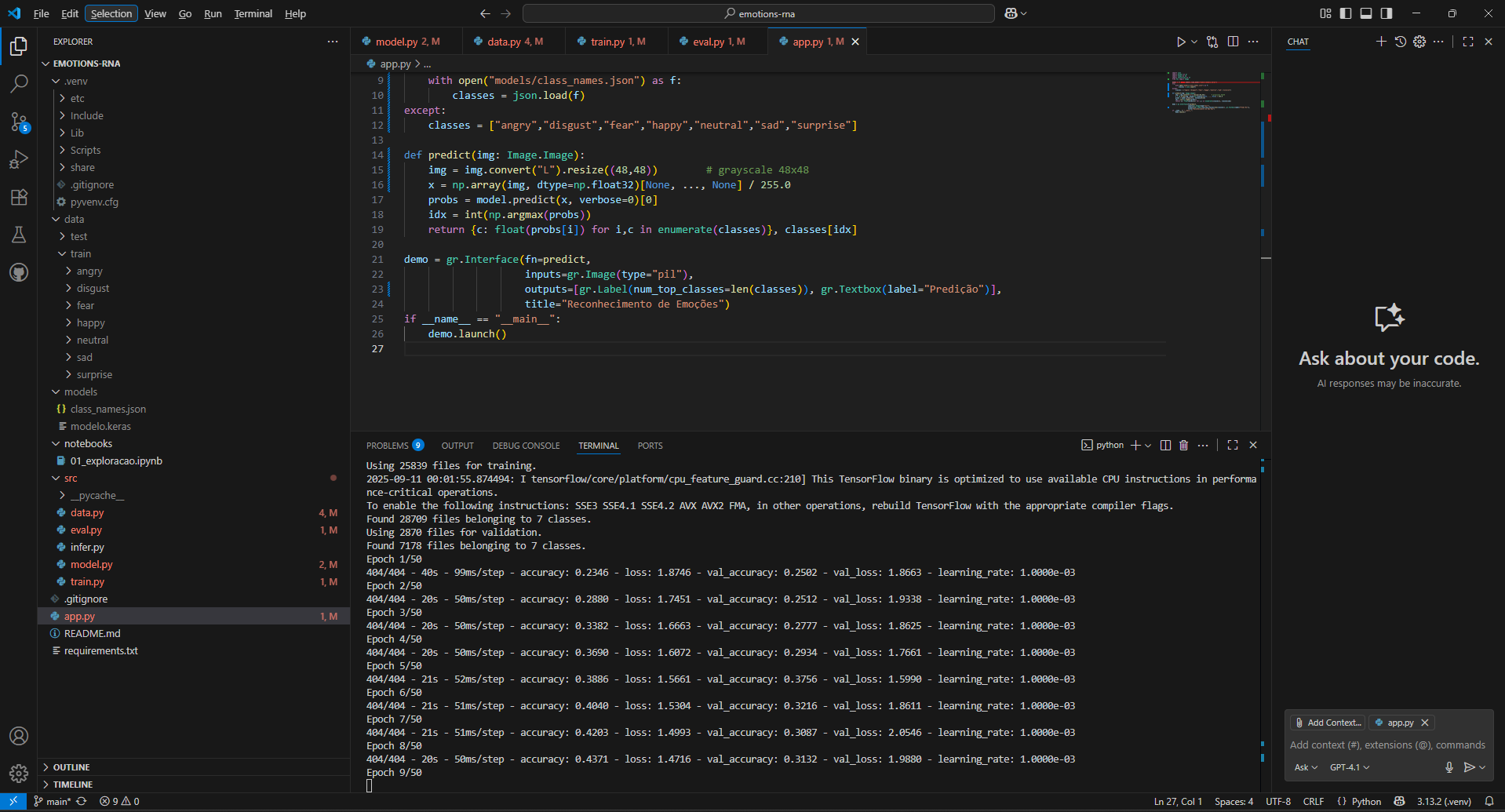
**Métricas e Avaliação**

* **Acurácia (Accuracy)**: Percentual de acertos totais.
* **Matriz de Confusão**: Tabela mostrando onde o modelo acerta e onde confunde as classes.
* **F1-score**: Média harmônica de precisão e revocação, útil quando as classes estão desbalanceadas.
* **Macro-F1**: Calcula o F1-score para cada classe e tira a média (trata todas as classes com igual importância).

**Conceitos Avançados**

* **Data Augmentation**: Técnica para “aumentar” os dados criando variações (rotacionar, espelhar, alterar brilho).
* **Transfer Learning**: Usar uma rede já treinada (ex.: MobileNet, EfficientNet) como ponto de partida para seu problema.
* **Regularização**: Estratégias para evitar *overfitting* (ex.: Dropout, L2, Early Stopping).
* **Early Stopping**: Interromper o treino quando a validação parar de melhorar.
* **Learning Rate (Taxa de Aprendizado)**: Define o “tamanho do passo” dado pelo otimizador ao atualizar os pesos.

Primeiro treinamento:



Implementando treinamento utilizando a GPU

