Q2 — Recognition (50 points)

Q2.1 — Task 1 (12 points)

Checklist (mark each satisfied item):

- Correct implementation.
- Model correctly flattens 28×28 input images to vectors of size **784**.
- Model contains at least one hidden layer **plus** one output layer.
- Hidden layers use ReLU (or other nonlinear) activations; output layer does **not** use a hidden activation.
- Model has **exactly 10** output units (for classification).
- Shows parameter-count calculation (code or manual work).
- Total model parameters **≤ 100,000**.
- Uses **SGD** or **Adam** optimizer.
- Uses **CrossEntropyLoss** for training.
- Records training loss and validation accuracy **once per epoch**.
- Implements early stopping: stops when validation accuracy does not improve for a defined patience.
- Validation accuracy **≥ 84%**.
- Reports training time (seconds).

> Scoring note: award points proportional to how many of the above items are satisfied, up to 12 total.

Q2.2 — Task 2 (8 points)

Checklist:

- Correct (follows instructions).
- Checks for GPU availability and creates appropriate `device` (e.g., `cuda` / `cpu`).
- Model is moved to GPU device when available.
- Records training loss and validation accuracy once per epoch.
- Uses identical MLP architecture as Task 1.
- Uses same loss, optimizer, and early-stopping scheme as Task 1.
- Validation accuracy **≥ 84%** (on GPU run).
- Reports GPU training time (seconds).
- Correctly computes speedup as `CPU time / GPU time`.

Q2.3 — Task 3 (12 points)

Checklist:

- Correct (follows instructions).
- Contains at least one `nn.Conv2d` layer with a nonlinear activation.
- Contains at least one pooling layer (`MaxPool2d` or `AvgPool2d`).
- Correctly flattens convolutional output before linear layers.
- Contains at least one hidden linear layer plus output layer after conv blocks.
- Exactly **10** output units for classification.
- Shows parameter-count calculation (manual or code).
- Total model parameters **≤ 100,000**.

- Trains model on GPU device.
- Uses a training loop with loss monitoring and early stopping.
- Uses **CrossEntropyLoss** for training.
- Validation accuracy **≥ 88%**.
- Reports CNN training time (seconds).

Q2.4 — Task 4 (18 points)

Checklist:

- Correct (follows instructions).
- Provides a function to evaluate the trained CNN from Task 3 **without modifying the model**.
- Evaluates performance on the **test_loader** (not train/validation).
- Reports baseline test accuracy (percentage) for the Task-3 model.
- Adds `nn.Dropout` layer in the correct location of the architecture.
- Uses dropout probability `p = 0.2`.
- Keeps CNN architecture identical to Task 3 except for the dropout addition.
- Trains the dropout-modified model with the same procedure as Task 3.
- Evaluates the dropout model on the **test_loader**.
- Correctly computes and reports the accuracy difference: `(with_dropout_accuracy - without_dropout_accuracy)`.

Q3 — Transfer (31 points)

Q3.1 — Task 1 (15 points)

Checklist:

- Correct (follows instructions).
- Sets `requires_grad = False` for all original ResNet parameters (using `model.parameters()`), i.e., parameter freezing done.
- Freezing occurs **before** any training and before replacing final layer.
- Replaces `model.fc` (or equivalent) with a `Linear` layer that has **exactly 10** output units.
- New `Linear` uses `model.fc.in_features` (or equivalent) for correct input dimension.
- Uses **CrossEntropyLoss** for training.
- Uses **SGD** or **Adam** optimizer.
- Implements early stopping based on validation accuracy with defined criteria (patience, etc.).
- Validation accuracy **≥ 40%**.
- Evaluates and reports test-set accuracy using `test_loader`.

Q3.2 — Task 2 (16 points)

Checklist:

- Correct (follows instructions).
- Starts from a pretrained **ResNet18** with default weights.

- Uses a **smaller learning rate** than Task 1 (fine-tuning typically uses a smaller lr).
- Replaces `.fc` with a `Linear` layer that has **exactly 10** outputs.
- **Does NOT freeze** any parameters — all model parameters are trainable.
- Uses **CrossEntropyLoss** for training.
- Uses **SGD** or **Adam** optimizer.
- Implements a proper training loop with loss computation and backpropagation.
- Records and monitors validation accuracy during training.
- Validation accuracy **≥ 74%**.
- Evaluates and reports test-set accuracy using `test_loader`.

Q4 — Interpretability & Accountability (19 points)

Q4.1 — Task 1 (10 points)

Checklist:

- Correct (answers the prompt).
- Provides a reasonable definition of **interpretable models**.
- Provides a reasonable definition of **explainable models**.
- Explicitly contrasts interpretable vs. explainable models and demonstrates understanding of the difference.
- Clearly states whether **Grad-CAM** saliency maps are helpful (or gives a nuanced/conditional answer).

- Gives **at least two** concrete reasons supporting their stance (examples: visual insight, bias detection, coarse localization, limited explanatory power, etc.).

- Shows awareness of Grad-CAM limitations (e.g., coarse spatial resolution, assumptions about final-layer semantics, “where” vs “why” distinction).

Q4.2 — Task 2 (9 points)

Checklist:

- Correct (answers the prompt).

- Takes a clear stance on government use of facial recognition (e.g., allow / ban / conditional/restricted).

- Presents logical reasoning with **at least two** supporting points for the stance.

- Acknowledges both potential benefits and the risks/concerns of the technology.

- Explicitly cites or references **at least two** of the provided sources (e.g., *Gender Shades* / Buolamwini & Gebru, Marks, Walsh).

- Incorporates specific facts, statistics, or examples from those sources (e.g., error-rate disparities, wrongful arrests, privacy violations).