IA - Clase 3A

(ML – Machine Learning)

Aprendizaje de Máquina

Ejercicio para comparar resultados con K-NN (784D y HOG), K-Means, GMM y CNN simple.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- ¿Qué significa entrenar un modelo?
  - Es el proceso de ajustar sus parámetros internos (pesos, coeficientes, centroides, etc.) para que aprenda a realizar una tarea, como clasificar letras en EMNIST.
  - Datos de entrada → imágenes 28×28 píxeles (tensores).
  - Modelo → puede ser una red neuronal, una regresión logística, un KNN, etc.
  - Parámetros → son números internos del modelo (ejemplo: pesos sinápticos en una red).
  - Objetivo → que, al darle una imagen, el modelo prediga la clase correcta (A, B, C, …).
  - Extended MNIST. Modified National Institute of Standards and Technology.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Etapas del entrenamiento
  - Inicialización.
    - El modelo arranca con pesos aleatorios o predefinidos.
  - Forward pass (predicción)
    - Se le pasa una imagen → el modelo genera una predicción (ej: cree que la letra es “C”).
  - Función de pérdida (loss) $\hat{P}$
- Compara la predicción del modelo contra la etiqueta real.
  - Ejemplo: predijo “C”
  - La etiqueta era “A”
- La pérdida mide cuán “mal” estuvo.
  - Fórmula típica:
  - Clasificación → se usa la entropía cruzada
  - Regresión → se usa el error cuadrático medio

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Backward pass (retropropagación)
  - El error se propaga hacia atrás para calcular cómo ajustar cada peso.
- Actualización de parámetros.
  - Se usa un optimizador (ej. gradiente descendente, Adam, SGD) que ajusta los pesos un poquito para reducir el error.
- Iteraciones (epochs)
  - Se repite el proceso sobre todo el dataset muchas veces hasta que el modelo “aprenda”.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Ejemplos de modelos:
  - Regresión logística multinomial (Softmax)
    - Logits: $z = Wx + b, \quad W \in \mathbb{R}^{26\times 784},\ x \in \mathbb{R}^{784},\ b \in \mathbb{R}^{26}$
    - Softmax (probabilidades): $p_k = \dfrac{e^{z_k}}{\sum_{j=1}^{26} e^{z_j}}, \quad k = 1, \ldots, 26$
    - Predicción: $\hat{y} = \arg\max_k p_k$
    - Pérdida (entropía cruzada, una muestra):

      $\mathcal{L} = -\sum_{k=1}^{26} y_k \log p_k$ (con y one-hot)
  - Perceptrón Multicapa (MLP) con 1 capa oculta
    - Aplanado: $x \in \mathbb{R}^{784}$
    - Capa oculta (ReLU):

      $h = \sigma(W_1 x + b_1), \quad W_1 \in \mathbb{R}^{m\times 784},\ b_1 \in \mathbb{R}^m,\ \sigma(u) = \max(0, u)$
    - Salida (logits y softmax):

      $z = W_2 h + b_2, \quad W_2 \in \mathbb{R}^{26\times m},\ b_2 \in \mathbb{R}^{26}, p_k = \dfrac{e^{z_k}}{\sum_{j=1}^{26} e^{z_j}}$
    - Pérdida: $\mathcal{L} = -\sum_{k=1}^{26} y_k \log p_k$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Logits son los valores crudos de salida de un modelo antes de aplicar una función de activación como sigmoid o softmax.
- Son los números reales (pueden ser negativos o positivos, incluso muy grandes) que la red neuronal calcula en la última capa lineal.
- Formalmente:
  - Si la última capa de una red es una transformación lineal
  - $z=wx+b$
    - Donde w son los pesos, x es el vector de entrada y b el sesgo, entonces z son los logits.
- No son probabilidades, porque no están en el rango [0,1].
- No necesariamente suman 1 en problemas multiclase.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- ¿Qué es softmax ?
  - Softmax toma un vector de puntajes (logits) $z \in R^K$ y lo convierte en una distribución de probabilidad sobre K clases

    1. **Exponentiar** cada puntaje: $s_i = e^{z_i}$ (todos quedan positivos).
    2. **Sumar**: $S = \sum_j s_j$.
    3. **Normalizar**: $p_i = s_i / S$ (cada $p_i \geq 0$ y $\sum_i p_i = 1$).

  - Los logits z_i son “afinidades” sin escala (pueden ser negativos, no son probabilidades).
  - La exponencial resalta diferencias: una clase con un logit un poco mayor recibe mucha más probabilidad.
  - La división por la suma convierte esos valores en una distribución (suma 1).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- ¿Qué es softmax ?
  - K fija la dimensión de salida y coincide con la cantidad de clases que se quiere predecir.
  - Es simplemente cuántas opciones hay. Si el problema es “¿gato, perro o zorro?” , entonces K = 3. Si es “¿qué letra de A a Z?”, entonces K = 26.
  - Softmax toma esa lista de puntuaciones y la convierte en porcentajes que son todos positivos, y suman 100% entre todas las opciones.
    Así “A: 5%, B: 20%, C: 75%”.
  - No cambia el orden: la opción con mayor puntuación cruda sigue siendo la más probable tras softmax.
  - Controla “confianza” (temperatura): se puede hacer la distribución más plana (más repartida) o más picuda (muy concentrada en la ganadora). Plana = más duda; picuda = más seguridad.

z_i: **logit** de clase i (salida lineal antes de softmax).
p_i: **probabilidad** de clase i después de softmax.
$e^{(\cdot)}$: exponencial, asegura no-negatividad.
$\sum_j e^{z_j}$: **normalizador** (suma a 1).
y_i: etiqueta **one-hot** (1 en la clase verdadera).
$\mathcal{L}$: **pérdida** (NLL / cross-entropy).
δ_{ij}: delta de Kronecker.
T: **temperatura** (suaviza o afila la distribución).
W, b: parámetros del modelo lineal que generan z
J: **Jacobiano** de softmax ($\mathrm{diag}(p) - pp^\top$).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- En Softmax usamos el nro de Euler e “exponencial” para convertir puntajes en algo que nunca sea negativo.
  - Porque hace el cálculo y las derivadas mucho más simples: $\frac{d}{dx}e^x = e^x.$
  - Nunca da números negativos.
  - Sea el número que sea (chico, cero o grande), al aplicarle la exponencial se obtienes un valor positivo.
  - Es como medir “brillo”: puede ser tenue o intenso, pero no existe el brillo negativo.
  - Todo puntaje aporta “algo” (aunque sea leve).
  - Un puntaje muy bajo no “resta”, solo aporta poco. Así evitamos resultados raros como “probabilidades negativas”.
  - Resalta al ganador sin romper el orden.
  - Si una opción tiene un puntaje un poco más alto, tras la exponencial se ve más clara la diferencia. Eso ayuda a elegir y, a la vez, mantiene el mismo ranking (la que era mayor sigue siendo mayor).
  - Fácil de convertir en porcentajes.
  - Como todos los valores quedan positivos, después es natural dividir por la suma y obtener números que suman 100% (probabilidades).
  - ¿Por qué la exponencial y no, por ejemplo, el cuadrado?
    - También daría positivos pero la exponencial hace que las diferencias pequeñas en los puntajes se vuelvan claras en los porcentajes y funciona bien con otras partes del entrenamiento.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- En Softmax usamos qué es la etiqueta y ?
  - Es la respuesta correcta para cada ejemplo del dataset.
  - Si se clasifican imágenes en K opciones (A..Z, por ejemplo), la etiqueta y dice cuál es la opción verdadera para esa imagen.
    - Como número (índice de clase) un solo entero que identifica la clase correcta.
      - Ej.: en letras A..Z, mapear A→1, B→2, …, Z→26.
      - Para una imagen de “T”, y=20.
    - Como vector “one-hot”.
      - Un vector con K posiciones: todo ceros salvo un 1 en la clase verdadera.
      - Ej.: si K=5 y la clase correcta es la 3, entonces y=[0, 0, 1, 0, 0]
  - Comparamos las probabilidades que da el modelo (el softmax) con la verdad y
  - Si el modelo pone mucha probabilidad en la clase correcta, premio (pérdida baja).
  - Si no, castigo (pérdida alta). Eso guía a la red a ajustar sus parámetros para acertar más.
    - Si el modelo dice: “A: 10%, B: 15%, C: 70%, D: 5%”.La etiqueta y (verdad) es “C”. Como el modelo ya puso el 70% en C, la comparación sale bien (poca corrección). Si hubiera dicho “B: 70%” y la verdad fuera “C”, la comparación sale mal y el modelo se corrige.
  - Label smoothing: en vez de poner un 1 perfecto en la clase correcta y 0 en el resto, se reparte un poco(por ej., 0.9 en la correcta y 0.1 repartido).
  - Ayuda a que el modelo no esté sobre-confiado y mejore su calibración.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- En Softmax explicamos las pérdidas:
  - Pérdida NLL (Negative Log-Likelihood)
    - Es un castigo que aplicamos al modelo según cuánta probabilidad le dio a la clase correcta.
    - Si el modelo dice “estoy muy seguro de la clase correcta”, el castigo es bajo.
    - Si le da poca probabilidad a la clase correcta, el castigo es grande.
  - Cómo se calcula:
    - El modelo da probabilidades por clase (softmax).
    - Analizo la probabilidad que asignó a la clase verdadera: $p_{verdadera}$
  - La pérdida es: NLL $= -\log(p_{verdadera})$
  - Por qué el “log” (intuición):
    - Si $p_{verdadera}$ es alta (p. ej., 0.9), $-\log(0.9)$ es chico.
    - Si es baja (0.1),$-\log(0.1)$ es grande.
    - Castiga mucho estar muy seguro y mal (mala calibración).
      - Ejemplo: s la verdadera es “C” y el modelo da p(C)=0.70
      - p(C)=0.70 → NLL ≈ 0.36.
      - Si da p(C)=0.10 p(C)=0.10 → NLL ≈ 2.30.
      - Más bajo es mejor
  - En un lote (batch): promediar la NLL de todos los ejemplos para tener una sola métrica.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- En Softmax explicamos las pérdidas:
  - Entropía Cruzada (Cross-Entropy)
    - Es la misma idea de “castigo” pero escrita de un modo que también sirve cuando la etiqueta no es un 1 perfecto sino que está suavizada (label smoothing) o hay varias clases verdaderas. Generaliza la NLL y se usa para comparar la verdad y con las probabilidades p del modelo (con o sin suavizado).
  - Análisis:
    - Tengo una “clase verdadera” y (normalmente one-hot: todo 0 salvo un 1 en la clase correcta).
    - Tengo probabilidades del modelo p (softmax).
    - La entropía cruzada mide cuánta incertidumbre hay cuando uso p para “codificar” la verdadera clase y
    - Cross-Entropy = “lo mal que describo la verdad con mis probabilidades”.
  - Relación con NLL (caso usual):
    - Si y es one-hot (una sola clase correcta),
    - Entropía Cruzada = NLL = $-\log(p_{\text{verdadera}})$
    - Si uso label smoothing (por ejemplo, 0.9 en la correcta y 0.1 repartido en las demás), la entropía cruzada sigue funcionando (NLL no alcanza porque asume un 1 perfecto).
  - Suave y estable para entrenar redes.
  - Calibra: empuja al modelo a asignar probabilidades razonables, no solo a acertar.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- En Softmax
  - δ_{ij} es la delta de Kronecker:
    - $\delta_{ij}=1$si $i=j$ (misma clase), $\delta_{ij}=0$ si $i\neq j$ (clases distintas).
    - interruptor: se enciende (1) solo cuando los índices coinciden, y se apaga (0) cuando no.
    - Calibra: empuja al modelo a asignar probabilidades razonables, no solo a acertar.
  - El Jacobiano de una función vectorial es la matriz de todas sus derivadas parciales. Para softmax, que transforma logits $z=(z_1,\dots,z_K)$ en probabilidades $p=(p_1,\dots,p_K)$, el Jacobiano dice cómo cambia cada probabilidad p_i si se ajusta un poquito algún logit z_j.
    - Entrada: un pequeño cambio en el input z_j
    - Salida: el efecto en el output p_i
    - Toda esa sensibilidad se organiza en una matriz K×K

$$\frac{\partial p_i}{\partial z_j} = p_i\left(\delta_{ij} - p_j\right) \quad \Longrightarrow \quad J \;=\; \mathrm{diag}(p) \;-\; p\,p^{\top}$$

- Si $i = j$: $\frac{\partial p_i}{\partial z_i} = p_i(1 - p_i)$ (subir su propio logit sube su probabilidad, pero con "tope" por $1 - p_i$).
- Si $i \neq j$: $\frac{\partial p_i}{\partial z_j} = -p_i\, p_j$ (subir el logit de otra clase le **quita** probabilidad a i).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- En Softmax
  - En el Jacobiano Diagonal (i=j):
    - “Si subo el control de la clase i, su probabilidad sube”.
    - Pero no puede crecer sin límite: el factor $1-p_i$ hace que el empuje sea menor cuanto más cerca está de 1 (saturación).
  - Fuera de la diagonal (i≠j):
    - “Si subo el control de otra clase j, le quito algo a i”.
    - ¿Cuánto le quito? Proporcional a $p_i\,p_j$: si ambas clases son relevantes, compiten más.
  - Dos propiedades útiles:
    - Cada fila suma 0 ($\sum_j \partial p_i/\partial z_j=0$) cambiar todos los logits por igual no cambia las probabilidades (invarianza a traslación).
    - Rango K−1: solo importan las diferencias de logits, no su valor absoluto.

Supongamos $p = [0.7,\ 0.2,\ 0.1]$ (tres clases).

- Diagonal:
  - $\partial p_1/\partial z_1 = 0.7(1-0.7) = 0.21$
  - $\partial p_2/\partial z_2 = 0.2(1-0.2) = 0.16$
  - $\partial p_3/\partial z_3 = 0.1(1-0.1) = 0.09$
- Fuera de la diagonal (negativos):
  - $\partial p_1/\partial z_2 = -0.7 \cdot 0.2 = -0.14$
  - $\partial p_1/\partial z_3 = -0.7 \cdot 0.1 = -0.07$
  - $\partial p_2/\partial z_1 = -0.2 \cdot 0.7 = -0.14$, etc.

- Interpretación: si se sube un poco z1, p1 sube ≈0.21×(es valor que sube z1), mientras p2 y p3 bajan en proporciones −0.14 y −0.07, respectivamente.
- El Jacobiano es la matriz que describe cómo reaccionan las probabilidades de todas las clases cuando se mueve un poco cada logit.
- El Jacobiano es la “brújula de sensibilidad”: te dice cómo ajustar los logits para que las probabilidades cambien en la dirección correcta durante el entrenamiento.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- K-NN
  - No hay pesos ni entrenamiento clásico.
  - El modelo se guarda todos los datos de entrenamiento (imágenes + etiquetas).
  - Para clasificar una nueva imagen: Se mide la distancia (usualmente Euclídea) entre esa imagen y cada imagen de entrenamiento.
  - Se eligen los k vecinos más cercanos.
  - Se predice la clase más frecuente entre esos vecinos (mayoría).
  - Distancia Euclídea entre el vector de entrada x y un ejemplo de entrenamiento x_i

$$d(x, x_i) = \sqrt{\sum_{j=1}^{784} (x_j - x_{i,j})^2}$$

  - Conjunto de índices de los k vecinos más cercanos

$$N_k(x) = \text{argmin}_{i=1,\ldots,n}^{k} \; d(x, x_i)$$

  - Predicción por mayoria de votos $\hat{y} = \text{mode}\{y_i : i \in N_k(x)\}$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- K-Means: objetivo de cuantización
  - Minimiza el error intra-cluster (distancia al centro).
  - Útil para compresión, prototipos e inicialización de GMM.
- Algoritmo de Lloyd
  - Paso E (asignación): asignar cada punto al centro más cercano.
  - Paso M (actualización): recomputar cada centro como media del cluster.
-

$$z_i = \operatorname{argmin}_{c \in \{1,\dots,K\}} \; |x_i - \mu_c|_2^2$$

$$\mu_c = \frac{1}{N_c} \sum_{i=1}^{N} x_i \, \mathbf{1}\{z_i = c\}, \quad N_c = \sum_{i=1}^{N} \mathbf{1}\{z_i = c\}$$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- K-Means: convergencia y clasificación
  - El coste desciende monótonamente hasta un mínimo local.
  - Clasificación: cluster→clase por mayoría (train).
  - Predicción = clase del centro más cercano.
  - Pseudo-probabilidades: normalizar inversos de distancias a centros.

$$J^{(t+1)} \leq J^{(t)}$$

$$\hat{y}(x) = \text{argmax}_y \sum_{i=1}^{N} \mathbf{1}\{y_i = y\}\, \mathbf{1}\{z_i = z^\star(x)\}$$

$$z^\star(x) = \text{argmin}_c \ |x - \mu_c|_2^2$$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- GMM: modelo y log-verosimilitud
  - Mezcla gaussiana: combinación convexa de gaussianas con pesos \(\pi_k\).
  - Parámetros: medias \(\mu_k\), covarianzas \(\Sigma_k\), pesos \(\pi_k\).

$$p(x) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k), \qquad \sum_{k=1}^{K} \pi_k = 1, \ \pi_k \geq 0$$

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \, \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- GMM: algoritmo EM
  - E-step: responsabilidades \(\gamma_{ik}\) (posterior del componente).
  - M-step: actualizar \(\mu_k\), \(\Sigma_k\), \(\pi_k\) ponderando por \(\gamma_{ik}\).

$$\gamma_{ik} = \frac{\pi_k \, \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}$$

$$N_k = \sum_{i=1}^{N} \gamma_{ik}, \quad \mu_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} \, x_i$$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- GMM: clasificación y relación con K-Means
  - Clasificación: componente→clase por mayoría (train); sumar responsabilidades por clase (test).
  - Límite: \(\Sigma_k=\sigma^2 I,\ \sigma\to 0\) ⇒ responsabilidades duras (K-Means).

$$\text{label}(k) = \text{argmax}_y \sum_{i=1}^{N} \mathbf{1}\{y_i = y\}\, \mathbf{1}\{\text{argmax}_j\, \gamma_{ij} = k\}$$

$$p(y = c \mid x) = \sum_{k\,:\,\text{label}(k)=c} \gamma_k(x), \quad \hat{y}(x) = \text{argmax}_c\ p(y = c \mid x)$$

$$\Sigma_k = \sigma^2 I,\ \sigma \to 0 \ \Rightarrow\ \gamma_{ik} \in \{0, 1\}$$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Selección de K (BIC)
  - En no supervisado: elegir K con BIC/AIC (penalización por complejidad).
  - Para EMNIST Letters como clasificador: K=26 por semántica

$$\mathrm{BIC} = \log L - \frac{p}{2}\log N, \quad p = K\left[d + \frac{d(d+1)}{2}\right] + (K-1)$$

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Clasificar un conjunto (set) de imagenes sencillas.
- Letras: quiero ingresar letras y que las clasifique en base a la base del EMINST (Extended MNIST) que es una extensión del dataset MNIST (dígitos manuscritos).
- Formato: cada imagen es en escala de grises, 28×28 píxeles, igual que MNIST.
- Etiquetas: números enteros que representan la clase (0–61 según la variante).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Variantes:
  - EMNIST ByClass: 814,255 caracteres, 62 clases (10 dígitos + 26 mayúsculas + 26 minúsculas).
  - EMNIST ByMerge: 814,255 caracteres, 47 clases (se fusionan mayúsculas y minúsculas similares, como C/c).
  - EMNIST Balanced: 131,600 caracteres, 47 clases (subconjunto balanceado).
  - EMNIST Letters: 145,600 caracteres, 26 clases (solo letras, sin distinción mayúscula/minúscula).
  - EMNIST Digits: 280,000 caracteres, 10 clases (solo dígitos).
  - EMNIST MNIST: 70,000 dígitos (idéntico al MNIST original, sirve para consistencia).
- Trabajaremos con EMNIST Letters (26 clases, 145 600 imágenes).
- Descarga del dataset
  - datasets.EMNIST(..., download=True) baja un archivo comprimido grande (cientos de MB).
  - Una vez descargado, queda guardado en ./data y ya no se vuelve a bajar.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Descompresión / preparación de archivos
  - EMNIST viene en formato IDX comprimido (.gz).
  - torchvision lo descomprime la primera vez.
  - Esta operación de descompresión y escritura suele tardar varios minutos (pero ocurre solo una vez).
- Construcción de índices internos
  - Al inicializar el objeto EMNIST, se abren los archivos IDX y se leen los encabezados (número de imágenes, resolución, etc.).
  - Luego de la primera vez en las ejecuciones siguientes no vuelve a descargar ni descomprimir.
  - Solo abre los archivos en disco. El acceso a cada imagen es inmediato porque se hace lazy loading (on-demand).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- TRAIN: Es el grupo de datos que el modelo ve para aprender.
  - Se usa para ajustar los parámetros del modelo (pesos en una red neuronal, centroides en K-means, etc.).
  - EMNIST Letters: train = 124 800 imágenes.
  - Cada imagen tiene su etiqueta (1–26).
- TEST: grupo separado, que el modelo no vio nunca durante el entrenamiento.
  - Se usa para medir qué tan bien generaliza el modelo a datos nuevos.
  - EMNIST Letters: test = 20 800 imágenes.
  - Tienen el mismo formato que las de train, pero distintas instancias manuscritas.
- Si entreno y evalúo con el mismo conjunto, el modelo podría memorizar (overfitting) y dar una falsa sensación de que funciona bien.
  - Con test, verifico que realmente entiende los patrones y no solo repitió lo visto.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Utilizamos datase EMNIST
  - Letters
- Objetivo: comparar K-NN (784D y HOG), K-Means, GMM y una CNN simple.
- Procesos para preprocesado de carácter manual (letra t):
  - umbrales, morfología, bbox, resize, centrado, deskew.
- Visualizaciones:
  - matriz de confusión, vecinos, y regiones K-NN en PCA-2D.
- Grilla de pruebas desde archivo (grid.json) y reportes automáticos.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Dataset: EMNIST Letters
  - 26 clases: A..Z (mayús/minús fusionadas).
  - Etiquetas 1..26 ↔ A..Z.
- Corrección de orientación: rotar –90° + espejo horizontal (solo para el dataset).
- Muestreo estratificado:
  - mismo nº de ejemplos por letra
  - evita UndefinedMetricWarning
- División típica: ~15–25k train, ~5–8k test.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Preprocesado del bitmap (28×28 estilo EMNIST)
- Convertir a escala de grises (L), auto-invertir si el fondo es claro (letra clara/fondo negro). Modo “L” = imagen en escala de grises de 8 bits (un solo canal, valores 0–255).
- Pillow (PIL) es la biblioteca estándar de procesamiento de imágenes en Python.
- Umbral adaptativo (ventana y offset) u Otsu para hallar bbox; padding asimétrico para no cortar la ‘t’.
  - Separar trazo (letra) de fondo para poderencontrar el bounding box (bbox) ajustado al trazo,recortar y luego redimensionar/padear sin perder partes finas (como el “palito” superior de la t).
  - Otsu (umbral global) busca un solo umbral que separe “fondo” y “trazo” mirando el histograma. Elige el umbral que maximiza la varianza entre clases (equivalente a separar lo más nítidamente posible dos montoncitos del histograma).
  - Cuándo usarlo: fondos bastante uniformes y buena iluminación.
  - Ventaja: simple y rápido.
  - Riesgo: si hay sombras o fondo irregular, puede cortar partes finas (ej. la cabeza de la t).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Umbral adaptativo (ventana y offset) u Otsu para hallar bbox; padding asimétrico para no cortar la ‘t’.
  - Convertido a L (grises).
  - Umbral: si el fondo es parejo, Otsu; si no, adaptativo (ventana impar + offset C).
  - Limpieza (componente más grande) y bbox por proyecciones.
  - Redimensionar el bbox para que el lado mayor ≈ 20 px.
  - Padding asimétrico hasta 28×28, reservando más arriba si la letra lo requiere (como la t).
  - Chequear que queden ≥2–4 px libres en todos los bordes y que no se corten ascendentes/descendentes.
- Librerías:
  - OpenCV (opencv-python): umbrales (Otsu/adaptativo), morfología, bbox, rotaciones, resize, momentos (para COM y orientación).
  - NumPy: arrays.
  - Pillow (PIL): para abrir/guardar y convertir a “L”; aunque OpenCV también lee/escribe.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Morfología: opening (quita ruido), closing (cierra cortes), dilatación horizontal leve (refuerza travesaño).
- Resize manteniendo aspecto (lado mayor=20) + ancho mínimo; centrar en 28×28; deskew con límites + recentrado por COM (Center of Mass)
  - Limpieza (opening): quita puntitos y suciedad alrededor del trazo.
  - Reparación (closing): cierra pequeños “cortes” en la letra para que se vea continua.
  - Refuerzo horizontal: una leve “engorda” solo en horizontal, para destacar travesaños (ej. la t).
  - Tamaño consistente: se escala la letra manteniendo su forma para que su lado mayor mida 20 píxeles; si queda demasiado angosta, se asegura un ancho mínimo para que no se vea “palito”.
  - Enmarcado estándar: la letra ya escalada se centra en un cuadro de 28×28 con márgenes parejos (como en EMNIST).
  - Enderezar (deskew) con límite: se corrige una inclinación leve para que la letra quede recta, sin pasarse.
  - Recentrado por COM (centro de masa): se ajusta su posición para que el “peso” visual de la letra quede en el centro del cuadro (como equilibrarla en una balanza).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- K-NN en 784D (baseline)
  - Característica: vector de 28×28 = 784 (intensidades normalizadas).
  - K vecinos, weights='distance' (vecinos cercanos pesan más).
  - Métrica: euclidean o cosine (más robusto a grosor de trazo).
  - Diagnóstico: vecinos más cercanos + votos por clase.
    - Una imagen 28×28 se aplana en una lista de 784 números (intensidades de píxeles).
    - K-NN clasifica buscando las K imágenes más parecidas (vecinas) en esa lista gigante de 784 datos.
    - “Parecidas” = distancia pequeña entre listas (tras normalizar).
    - La etiqueta final es la más votada por esos K vecinos.
  - Pros: muy simple, no “entrena” modelos complejos.
  - Contras: en 784 dimensiones las distancias se vuelven menos claras y el método puede ser lento con muchos ejemplos (a veces se reduce la dimensión con PCA o se usan índices rápidos).
  - Metáfora: mostrás una letra nueva y el sistema dice: “se parece a estas 5; la mayoría son ‘A’, así que es A”.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Visualizaciones para K-NN
  - Matriz de confusión (test): Clases más confundidas (p. ej., T↔L/F).
- Vecinos más cercanos: inspección cualitativa de la decisión.
- Regiones K-NN (PCA-2D): solo visual para entender la frontera.
  - Tenés datos con muchas variables (p. ej., una imagen 28×28 = 784 números).
  - PCA busca las direcciones donde los datos varían más (las “tendencias principales”).
  - Luego proyecta todo a 2 dimensiones (2 ejes nuevos) que conservan la mayor información posible.
  - Resultado: un mapa 2D donde puntos parecidos quedan cerca y distintos, lejos. Sirve para ver grupos a simple vista.
  - Metáfora: imaginá una escultura iluminada; PCA elige el ángulo de luz que hace la sombra más informativa en una pared 2D.
  - Para letras (EMNIST): cada letra (784D) se comprime a 2D para visualizar: las “A” tienden a agruparse, separadas de las “B”/“C”, etc.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Apretamos 784 datos en 2: al comprimir tanta información a 2 ejes, distintos casos caen en el mismo lugar. Es normal que se superpongan.
- PCA no “mira” las etiquetas: elige direcciones con más variación, no las que mejor separan clases; por eso clases distintas pueden quedar encimadas.
- Letras parecidas y escrituras variadas: “O/0”, “I/1/l”, “c/e”, trazos finos/gruesos, inclinación, centrado... todo eso acerca puntos de clases distintas.
- Ruido y preprocesado: si no se deskew/centra/normaliza bien, la nube se ensancha y mezcla.
- Saturación visual: hay muchos puntos; sin transparencia/tamaño pequeño, la vista se vuelve “masa” de color.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Reportes: se guardan PNG y JSON/CSV con votos/predicciones.
- UndefinedMetricWarning: por qué aparece y cómo evitarlo
  - No se puede calcular bien algún indicador (precisión/recuperación) porque una clase no aparece en las pruebas o el modelo jamás la eligió.
  - Hay clases sin ejemplos o sin aciertos, y los cálculos se quedan “sin denominador”.
  - ¿Por qué pasa?
    - En el conjunto de prueba no cayó ninguna muestra de cierta clase.
    - El modelo nunca predijo esa clase (ni una vez).
  - Cómo evitarlo (versión simple):
    - Partir los datos con estratificación: al dividir entreno/prueba, garantizá que todas las clases estén representadas en ambos (train_test_split con stratify).
    - Aumentar cobertura de clases: más ejemplos de las clases raras o balancear datos (colectar más, equilibrar).
    - Ajustar el modelo: cambia K (en K-NN), la métrica de distancia o el umbral para que el modelo sí elija esas clases alguna vez.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Reportes: se guardan PNG y JSON/CSV con votos/predicciones.
- UndefinedMetricWarning: por qué aparece y cómo evitarlo
  - Silenciar el cero (solo para el informe): usar zero_division=0 en el reporte para que, cuando falten casos, muestre 0 en lugar de explotar.
  - Idea clave: asegurá muestras y predicciones para todas las clases; si una clase “no existe” en prueba o nunca se predice, los indicadores para esa clase quedan indefinidos.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- K-NN con HOG (mejor para encontrar ‘t’ vs ‘l’)
  - HOG para 28×28: orientaciones=9, celdas 4×4, bloques 2×2, L2-Hys.
  - Captura bordes/direcciones: el travesaño de ‘t’ deja firma horizontal.
  - K-NN sobre HOG suele mejorar frente a intensidades puras (784D).
  - Usar K-Nearest Neighbors no sobre píxeles crudos, sino sobre descriptores HOG (Histogram of Oriented Gradients), que resumen bordes y direcciones de la imagen.
  - HOG convierte cada letra en un vector “de contornos”; K-NN compara esos vectores y clasifica por vecinos más parecidos.
    - Preprocesar: pasar a gris “L”, binarizar/adaptativo, deskew, centrar por COM, escalar a 28×28.
    - Calcular HOG por imagen (p. ej. 9 orientaciones, celdas 4×4 px, bloques 2×2, norma L2-Hys).
    - Estandarizar los vectores (StandardScaler).
    - Entrenar K-NN (elige K y métrica: euclídea, coseno, etc.).
    - Validar con estratificación y ajustar K/params HOG.
    - Reportar métricas (usa zero_division=0 para evitar warnings si falta alguna clase).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Desambiguador L↔T (post-proceso)
  - 1) Votos K-NN: si P(T) ≈ P(L) (margen pequeño),
  - 2) ‘Crossbar score’: energía horizontal en banda superior-central.
  - Si score ≥ τ y P(T) no muy inferior a P(L), reasignar L→T.
  - Parámetros típicos: τ≈1.05–1.20, margen≈0.05–0.10 (ajustar al trazo).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Grilla de pruebas (grid.json) + reportes
  - Archivo externo con experimentos: modelo (784D/HOG), K, métrica, umbrales, dilatación…
  - Cada experimento: genera bitmap_postproc.png, vecinos.png, regiones.png y report.json.
- resumen.csv con los resultados de la grilla.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- K-Means (no supervisado) como clasificador
  - Entrenar con 26 clusters (A..Z).
  - Mapear cluster→clase por mayoría en train (pseudoetiquetado).
  - ‘Probabilidades’ por clase a partir de 1/distancia a centroides (normalizada).
  - Útil para explorar estructura; suele rendir menos que K-NN/CNN.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- GMM (Gaussian Mixture) como clasificador
  - Entrenar con 26 componentes (full covariance).
  - Mapear componente→clase por mayoría; usar responsabilidades como prob.
  - Suaviza fronteras; puede capturar subformas (multi-modos) por letra.
  - Visualización: regiones en PCA-2D (solo visual).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Conv(1→16): simula 16 detectores que “pasean” por la imagen en blanco y negro (1 canal).
- Cada detector aprende a reconocer rasgos simples: bordes, curvas, cruces. (Tras cada conv suele ir una activación tipo ReLU: deja pasar lo importante).
- MP (Max-Pooling): una reducción del tamaño que se queda con lo más “fuerte” de cada zona. Así la imagen se resume y el modelo se vuelve más robusto a pequeños desplazamientos.
- Conv(16→32): ahora hay 32 detectores que combinan los rasgos simples para formar patrones más complejos (por ejemplo, la barriguita de una “a” o el travesaño de una “t”).
- MP: otra reducción para seguir compactando la información y el ruido.
- FC(64): se “aplana” todo y se pasa por una capa densa con 64 neuronas: es como un resumen numérico de la letra, listo para decidir.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Softmax(26): transforma ese resumen en 26 probabilidades (una por cada letra). La más alta es la predicción.
- Idea general: la red primero detecta piezas pequeñas (bordes), luego armados más complejos (partes de letras), y al final decide qué letra es. Es chica, rápida y funciona bien para letras centradas y normalizadas (tipo EMNIST).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- CNN simple (PyTorch)
  - Arquitectura: Conv(1→16)–MP–Conv(16→32)–MP–FC(64)–Softmax(26).
- Entrenar 5–15 épocas (estratificado ayuda).
- Evaluar: accuracy, matriz de confusión, top-8 probabilidades para tu bitmap.
- Regiones 2D: usar features de la penúltima capa + KNN 2D para visual.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Comparativa y recomendaciones
  - 784D vs HOG: HOG suele ganar en t/l/f y otros pares confusos.
  - K-Means/GMM: buenos para explorar; asignación por mayoría necesaria.
  - CNN: mejor performance si se ejecutan más épocas y muchos datos.
- Afinar preprocesado: adaptive offset, dilatación horizontal, ancho mínimo, deskew limitado.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- La matriz de confusión es una tabla que se utiliza para evaluar el rendimiento de un modelo de clasificación.
- Resume en forma de cuadrícula qué tan bien las predicciones del modelo coinciden con las clases reales.
- Ejes:
  - Filas = clases reales (lo que en verdad eran los datos).
  - Columnas = clases predichas (lo que dijo el modelo).
  - Diagonal principal (↘):
    - Muestra los aciertos (ej.: clase A real predicha como A).
    - Cuanto más fuerte el color en la diagonal → mejor el modelo.
  - Fuera de la diagonal:
    - Son los errores de clasificación (ej.: clase B real pero el modelo dijo C).
    - El color aquí indica confusión entre clases.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Resumen por clase con métricas:
  - Precision: de los que predijo como esa clase, ¿cuántos realmente lo eran?
  - Recall: de los que realmente eran de esa clase, ¿cuántos detectó?
  - F1: balance entre precision y recall.
    - Esto permite ver si el modelo está equilibrado.
- Ejemplo típico en reconocimiento de letras/dígitos:
  - “0” puede tener F1=0.98 (muy bien clasificado).
  - “8” puede tener F1=0.72 (se confunde con 3 y 0).
- Nos orienta a dónde mejorar datos o arquitectura.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Gráfica en PC1–PC2 (proyección PCA)
- Reducción de dimensionalidad para visualizar los datos y las predicciones:
- Qué significa:
  - Se toman imágenes/vectores de alta dimensión (ej. 784 píxeles de MNIST)
  - Se aplica PCA → se proyectan en solo 2 componentes principales (PC1 y PC2) que concentran la mayor varianza.
  - Cada punto es un ejemplo, coloreado según su clase real o predicha.
- Cómo se interpreta:
  - Agrupamientos o nubes de puntos: si una clase forma un grupo separado, significa que sus características son distintas y fáciles de clasificar.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Zonas mezcladas: si dos clases aparecen superpuestas (ej. dígitos “4” y “9”), indica que se parecen mucho en las características → el modelo puede confundirlos.
- Colores en el mismo grupo: si los colores corresponden a etiquetas predichas, podemos ver dónde el modelo se equivocó (ej.: puntos de clase “3” pintados con color de “8”).
- Es una herramienta exploratoria y visual, no exacta. Nos ayuda a entender por qué aparecen confusiones en la matriz de confusión.
- ¿Qué son PC1 y PC2?
  - Son las dos primeras Componentes Principales obtenidas por PCA (Análisis de Componentes Principales):
    - PC1 (Primer Componente Principal):
    - Es la dirección (eje) en el espacio de datos donde hay mayor variabilidad.
    - Resume la característica más importante que distingue los datos.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- PC2 (Segundo Componente Principal):
- Es la segunda dirección de máxima variabilidad, perpendicular a PC1.
- Resume la segunda característica más importante e independiente de la primera.

- Cómo se obtienen
  - datos en alta dimensión (ej. imágenes $28 \times 28 \rightarrow 784$ píxeles).
  - PCA calcula combinaciones lineales de esas variables (una especie de mezcla ponderada de píxeles).
  - PC1 es la combinación que explica la mayor parte de las diferencias entre ejemplos.
  - PC2 explica la segunda mayor parte, sin solaparse con PC1.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Si tuviéramos frutas medidas por peso y color:
  - PC1 podría captar la variación “peso” (frutas grandes vs chicas).
  - PC2 podría captar la variación “color” (verde vs rojo).

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- Interpretación en la gráfica PC1–PC2
  - Cuando graficamos los datos en 2D:
  - Eje horizontal = PC1
  - Eje vertical = PC2
  - Cada punto es un ejemplo transformado a ese espacio reducido.
  - Si las clases forman grupos separados, significa que en esas direcciones (PC1, PC2) se distinguen bien.
  - Si aparecen mezcladas, quiere decir que los rasgos principales no alcanzan para separarlas fácilmente.
- Los ejes (PC1 y PC2) resumen variaciones de los datos.
  - Los colores indican a qué clase pertenece cada ejemplo (real o predicha).
  - La mezcla de colores en la gráfica explica visualmente las confusiones que después vemos en la matriz de confusión.

Aprendizaje de Máquina (ML)
Ejercicio para comparar modelos

- PC1 captura si la letra es “cerrada” (ej. O, D, 0) o “abierta” (ej. L, I).
- PC2 captura si tiene “curvas” (ej. S, 8) o “rectas” (ej. H, T).
- Al graficar:
  - Puntos amarillos = letra O → nube redondeada.
  - Puntos verdes = letra I → nube distinta.
  - Si vemos mezcla de verde y amarillo, significa que hay manuscritos de “O” que parecen “I” al modelo.