UNIVERSIDAD TECNOLOGICA CENTROAMERICANA

**UNITEC**

Imagen que contiene Icono

Descripción generada automáticamente

**Sistemas Inteligentes**

**5:20 pm – 6:40 pm**

**PARCIAL II**

**ESTUDIANTE:**

Edgar Ricardo Ayala Polanco

**DOCENTE**:

Ing. Iván De Jesús Deras Tabora

**TITULO:**

Proyecto de Sistemas Inteligentes

**FECHA:** 27 / 3 / 2025

**LUGAR:** Campus San Pedro Sula

**Metodología**

Arquitectura de la Red

La red neuronal implementada es multicapa con la siguiente estructura:

* *Entrada:* 784 neuronas (imágenes MNIST de 28x28 píxeles).
* *Capas Ocultas:*
  + Capa 1: 128 neuronas, activación ReLU.
  + Capa 2: 64 neuronas, activación ReLU.
* Salida: 10 neuronas (Softmax para clasificación).

**2. Implementación**

Componentes Principales

* *Clase DenseLayer:* Capa totalmente conectada con propagación hacia adelante y atrás, soporte para L2 y guardado/carga de pesos.
* *Activaciones:*
  + ReLU en capas ocultas.
  + Softmax en la salida.
* *Función de Pérdida:* Entropía Cruzada con gradientes optimizados.
* *Optimizador:* Adam.

**3. Entrenamiento**

* *Inicialización:* Pesos, y opción de cargar modelos previos.
* *Proceso:*
  + Mezcla aleatoria en cada época.
  + Paso hacia adelante y cálculo de pérdida.
  + Retropropagación y actualización con Adam.
* *Regularización:* L2 (λ=0.0001).

**4. Monitoreo y Eficiencia**

* *Seguimiento:* Registro de pérdida y precisión en cada época.
* *Optimización:* Operaciones vectorizadas con NumPy, procesamiento en lotes.
* *Características* Prácticas: Guardado/carga de pesos, visualización de métricas y predicciones.

**5. Hiperparámetros**

* Tasa de aprendizaje: 0.1 (Adam con decaimiento 0.001).
* Épocas: 10
* β1: 0.9 (momento)
* β2: 0.999 (RMSprop)
* ε: 1e-7 (estabilidad numérica)

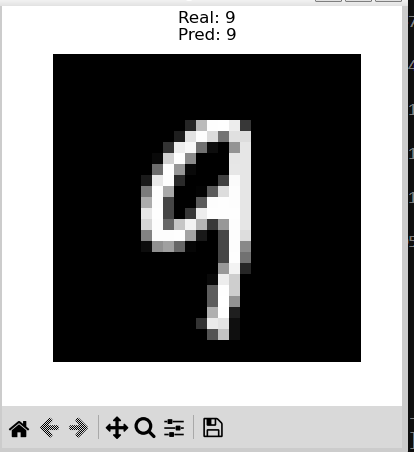
**Resultados código base**

Configuración predeterminada:

**Texto

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Aplicación, Código QR

El contenido generado por IA puede ser incorrecto.**

**Código QR

El contenido generado por IA puede ser incorrecto.**

**Inicio: Final:**

**Imagen que contiene exterior, lado, estacionado, tabla

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.**

**Experimentos (Hiperparametros)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EPOCHS | BATCH\_SIZE | SIZE1 | SIZE2 | LEARNING\_RATE | INITIAL ACCURACY | FINAL ACCURACY | INITIAL LOSS | FINAL LOSS |
| 20 | 32 | 64 | 32 | 0.1 | 80.01% | 84.71% | 0.8258 | 0.5358 |
| 20 | 32 | 128 | 64 | 0.1 | 58.80% | 82.35% | 1.3204 | 0.6035 |
| 20 | 64 | 128 | 64 | 0.01 | 82.19% | 83.17% | 0.6103 | 0.5762 |
| 20 | 32 | 64 | 64 | 0.01 | 94.87% | 97.27% | 0.3097 | 0.0099 |
| 20 | 64 | 128 | 64 | 0.001 | 95.00% | 97.75% | 0.3509 | 0.018 |
| 20 | 64 | 64 | 32 | 0.001 | 94.41% | 97.31% | 0.3494 | 0.0547 |
| 15 | 64 | 64 | 32 | 0.01 | 95.80% | 97.82% | 0.1804 | 0.0028 |
| 15 | 32 | 128 | 32 | 0.1 | 77.88% | 94.03% | 0.9268 | 0.1848 |

**Resultados**

1. *Mejor Configuración:*
   * BATCH=64, SIZE1=64, SIZE2=32, LR=0.01 (15 épocas).
   * Precisión: 97.82% | Pérdida: 0.0028.
2. *Learning Rate:*
   * Alto (0.1): Inestable (58.8% → 82.4%).
   * Bajo (0.001): Lento pero preciso (95% → 97.8%).
3. *Batch Size:*
   * 32: Más fluctuaciones.
   * 64: Entrenamiento estable.
4. *Arquitectura:*
   * Redes pequeñas (64-32) superaron a grandes (128-64) en MNIST.

**Desafíos**

*Problema:*

* Tenía lambda\_l2 declarado como parámetro en múltiples funciones (backward en DenseLayer, train, etc.), lo que generaba inconsistencia en su valor.
* No estaba claro si debía pasarse como argumento o definirse globalmente.

*Solución:*

* Unificación del parámetro: Se definió lambda\_l2 como una constante en main.py y se pasó consistentemente a todas las funciones que lo requerían.

*Problema:*

* Errores frecuentes al calcular gradientes debido a dimensiones incorrectas en matrices (ej: dweights con forma distinta a weights).
* Problemas al propagar grad\_output entre capas, especialmente después de Softmax.

*Solución:*

* Verificación de formas: Se implementaron chequeos con .shape en DenseLayer.backward().
* Transposición correcta de matrices: Se ajustó np.dot(inputs.T, grad\_output) para garantizar que las dimensiones coincidan**.**

*Problema:*

* Errores silenciosos en backpropagation (ej: gradientes incorrectos, pero sin mensajes de error).

*Solución:*

* Logging de pesos y gradientes:
  + Impresión de pesos antes/después del entrenamiento (primeras 5x5 elementos).
  + Monitoreo de pérdida y precisión por época.