Evaluación comparativa de arquitecturas CNN en la clasificacion de 83 variedades de papas

Sebastian Victor Torreblanca Paz
Escuela Profesional de Ingeniería Informática y de Sistemas
Universidad Nacional San Antonio Abad del Cusco
204808@unsaac.edu.pe

Milton Alexis Pachari Lipa
Escuela Profesional de Ingeniería Informática y de Sistemas
Universidad Nacional San Antonio Abad del Cusco
alexispacharii@gmail.com

Boris Eloy Sullcarani Diaz
Escuela Profesional de Ingeniería Informática y de Sistemas
Universidad Nacional San Antonio Abad del Cusco
200788@unsaac.edu.pe

Resumen: Este estudio evalúa comparativamente cuatro arquitecturas CNN: la red personalizada *PapaNet*, VGG16, SqueezeNet y MobileNetV2 para clasificar 83 variedades de papas nativas peruanas usando imágenes de 128×128 píxeles. Se analizó el impacto de hiperparámetros clave como tasas de aprendizaje, tamaño de lote y patrones de filtros, además de técnicas de regularización como BatchNorm y Dropout progresivo. MobileNetV2 obtuvo el mayor accuracy (91.67%), seguido de SqueezeNet (91.16%), destacando su eficiencia computacional con 2.36 millones de parámetros y 49 ms por lote. La aplicación de data augmentation con rotaciones, ajustes de brillo y flips horizontales amplió el dataset y mejoró la generalización. El modelo *PapaNet* mostró un desempeño competitivo con 84.98% de precisión y diseño ligero (1.3 millones de parámetros), mientras que VGG16 presentó menor eficiencia con 85.24% y 18.9 millones de parámetros. La optimización se realizó con AdamW, logrando entrenamientos estables.

Palabras Clave: deep learning, CNN, regularización, data augmentation, clasificación de papas, hiperparámetros, MobileNetV2

***Abstract*—This study presents a comparative evaluation of four CNN architectures: a custom *PapaNet*, VGG16, SqueezeNet, and MobileNetV2 for classifying 83 native Peruvian potato varieties using 128×128 px images. Key hyperparameters such as learning rates, batch size and filter patterns were analyzed, along with regularization techniques including BatchNorm and progressive Dropout. MobileNetV2 achieved the highest accuracy (91.67%), followed by SqueezeNet (91.16%), highlighting computational efficiency with 2.36 million parameters and 49 ms per batch. Data augmentation with rotations, brightness adjustments, and horizontal flips expanded the dataset and improved generalization. The *PapaNet* model showed competitive performance with 84.98% accuracy and a lightweight design (1.3 million parameters), while VGG16 showed lower efficiency with 85.24% accuracy and 18.9 million parameters. Optimization was performed using AdamW, achieving stable training.**

***Index Terms*—deep learning, CNN, regularization, data augmentation, potato classification, hyperparameters, MobileNetV2**

I. Introducción

Este estudio aborda el desafío de clasificar automáticamente las 83 variedades de papas nativas peruanas mediante cuatro arquitecturas CNN (una red personalizada y los modelos preentrenados VGG16, SqueezeNet y MobileNetV2), evaluando específicamente su accuracy en imágenes de 128 × 128 px. El análisis incluye la optimización de hiperparámetros clave como tasa de aprendizaje, batch size y patrones de filtros (32-128 por capa), junto con su impacto en la convergencia y eficiencia computacional. Adicionalmente, se validan técnicas de regularización (BatchNorm y Dropout) para estabilizar el entrenamiento, mientras que técnicas de data augmentation complementan el procesamiento.

Nuestras contribuciones principales son:

- esarrollamos una CNN especializada que detecta eficientemente las diferencias entre variedades de papa, optimizando precisión y rendimiento para uso en campo.
- Realizamos una comparación detallada entre modelos preentrenados y nuestra red, midiendo precisión, velocidad y uso de memoria para guiar la elección del mejor modelo según necesidades prácticas.
- Optimizamos sistemáticamente los parámetros clave: tasas de aprendizaje, tamaños de lote, configuraciones de filtros y técnicas de regularización para lograr el mejor rendimiento en el entrenamiento.

II. Marco Teórico

A. *Fundamentos de CNN para Visión Computacional*

Las Redes Neuronales Convolucionales (CNN) son particularmente efectivas para clasificación de imágenes agrícolas debido a tres propiedades clave:

- **Aprendizaje jerárquico**: Detectan progresivamente bordes, texturas y patrones complejos mediante filtros convolucionales
- **Invarianza espacial**: Mantienen rendimiento ante variaciones de posición y orientación
- **Eficiencia paramétrica**: Reducen parámetros mediante convoluciones y pooling

Para clasificación de 83 variedades de papas, enfrentamos:

$$\mathcal{L} = -\sum_{c=1}^{83} y_c \log(p_c) + \lambda \sum_i w_i^2 \tag{1}$$

donde p_c es la probabilidad predicha para la clase c y λ controla la regularización L2.

B. *Arquitecturas Comparadas*

1) Modelo Personalizado ("PapaNet"): Diseñado específicamente para el problema:

- **Patrón de filtros**: Ascendente (64→128→256) para capturar jerarquías de características
- **Regularización**: Combinación de Dropout progresivo (0.3→0.5) y L2 ($\lambda = 10^{-4}$)
- **Pooling**: Global Average Pooling reduce parámetros:

$$p_{ij} = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{hwij} \tag{2}$$

- **Ventajas**: 1.3M parámetros, 84.98% accuracy, 61ms/batch

2) VGG16 con Transfer Learning: Adaptación del modelo clásico:

- **Extractor de características**: 13 capas convolucionales preentrenadas en ImageNet
- **Fine-tuning**: Solo últimas 3 bloques entrenables
- **Regularización**: Dropout (0.5) en capas fully-connected
- **Desventaja**: 18.9M parámetros (72MB), 85.24% accuracy

3) SqueezeNet Optimizado: Arquitectura eficiente con:

- **Fire Modules**:
  - *Squeeze*: Convolución 1x1 reduce canales
  - *Expand*: Convoluciones 1x1 y 3x3 en paralelo
  - *Concatenación*: Fusiona ambas salidas
- **Ventajas**: Solo 0.77M parámetros (2.96MB), 91.16% accuracy
- **Ecuación clave**:

$$\begin{aligned} \text{Fire}(x) = [&\sigma(W_{1x1} * \sigma(W_{\text{squeeze}} * x)) \\ &\parallel \sigma(W_{3x3} * \sigma(W_{\text{squeeze}} * x))] \end{aligned} \tag{3}$$

4) MobileNetV2: Arquitectura eficiente con:

- **Bloques Invertidos Residuales**:
  - *Depthwise Separable Convolutions*: reducen cómputo y parámetros
  - *Atajos residuales*: mejoran el flujo del gradiente
  - *Bottleneck con expansión y proyección*: más eficiencia
- **Ventajas**: 2.36M parámetros (9.02MB), 91.67% de accuracy en validación final, entrenamiento total: 19.96 minutos
- **Ecuación clave**:

$$\text{Bloque}(x) = x + \text{PW}_{\text{linear}}(\text{DW}(\text{ReLU}(\text{PW}_{\text{expand}}(x)))) \tag{4}$$

TABLE I: Comparación de Arquitecturas

Característica	PapaNet	VGG16	SqueezeNet	MobileNetV2
Parámetros (M)	1.3	18.9	0.77	2.36
Accuracy (%)	84.98	85.24	91.16	91.67
Tiempo/batch (ms)	61	108	41	49
Memoria (MB)	5.06	72.30	2.96	9.02

C. *Comparativa Teórica*

D. *Fundamentos Matemáticos*

1) Batch Normalization: Técnica esencial para entrenar redes profundas de manera estable. Normaliza las activaciones en cada lote mediante:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} * \gamma + \beta \tag{5}$$

donde μ_B y σ_B son la media y varianza del lote, γ y β parámetros aprendibles, y $\epsilon = 10^{-5}$ para estabilidad numérica. En nuestro estudio:

- Permite tasas de aprendizaje 5-10 veces mayores
- Reduce el sobreajuste actuando como regularizador
- Es crucial para entrenar nuestra red personalizada

2) Optimizador AdamW: Versión mejorada de Adam que maneja mejor la regularización L2:

$$\theta_t = \theta_{t-1} - \eta \frac{m_t}{\sqrt{v_t} + \epsilon} - \eta \lambda \theta_{t-1} \tag{6}$$

Configuración clave:

- $\eta = 3 \times 10^{-4}$ (tasa de aprendizaje inicial)
- $\lambda = 10^{-4}$ (peso de la regularización)
- $\beta_1 = 0.9$, $\beta_2 = 0.999$ (momentos)

Ventajas para nuestro problema:

- Convergencia más rápida que SGD estándar
- Manejo automático de tasas de aprendizaje por parámetro
- Ideal para combinar con BatchNorm

3) Data Augmentation: Transformaciones aplicadas para mejorar la generalización:

$$\mathcal{T}(x) = \{\tau(x) | \tau \in \text{Rotación}(\pm 15), \text{Flip}(0.5), \text{Brightness}(\pm 0.2)\} \tag{7}$$

Impacto en el dataset:

- +300% de imágenes efectivas para entrenamiento
- Mejora de 4-6% en accuracy de validación
- Especialmente útil para clases minoritarias

III. METODOLOGÍA

A. *Dataset y Preprocesamiento*

Dataset: Utilizamos un conjunto original de 83 variedades de papas nativas peruanas, con imágenes en resolución 128×128×3 píxeles. El dataset fue recolectado en condiciones controladas de iluminación y fondo, garantizando la calidad requerida para el análisis.

Preprocesamiento: Implementamos las siguientes etapas:

- **Redimensionamiento:** Estandarización a 128×128 píxeles usando interpolación bicúbica

- **Data Augmentation:** Generación de imágenes adicionales mediante:
  - Rotaciones controladas (±15°)
  - Ajustes de brillo/contraste (±20%)
  - Flip horizontal aleatorio
- **Balanceo:** Aumento a 300 imágenes por clase mediante las transformaciones anteriores

Normalización: Aplicamos estandarización (Z-score) con valores calculados del conjunto de entrenamiento:

- Media: [0.485, 0.456, 0.406]
- Desviación estándar: [0.229, 0.224, 0.225]

B. Configuración Experimental

Entorno:

- Hardware: GPU T4 de Google Colab (16GB RAM)
- Framework: TensorFlow 2.18.0
- Semilla aleatoria fija (42) para reproducibilidad

Partición de datos:

- 70% entrenamiento
- 15% validación
- 15% prueba
- Estratificación por clase para mantener proporciones

C. Arquitecturas CNN

Red Personalizada ("PapaNet"):

- 6 capas convolucionales con filtros ascendentes (32→128).
- Global Average Pooling en lugar de capas fully-connected.
- Batch Normalization después de cada capa convolucional.
- Dropout (0.3-0.5) para regularización.
- GlobalAveragePooling en vez de capas fully connected.
- Capa densa de 512 neuronas y 83 para clasificar.

Modelos Preentrenados:

- **VGG16:** Fine-tuning en últimos 3 bloques
- **ResNet50:** Congelamiento parcial (primeras 40 capas)
- **EfficientNetB0:** Ajuste completo con learning rate reducido

D. Entrenamiento y Evaluación

Hiperparámetros:

- Batch size: 64 (óptimo determinado experimentalmente)
- Learning rate: 1e-4 a 1e-2 con decay coseno
- Épocas máximas: 40 (con early stopping)
- Función de pérdida: Categorical Crossentropy
- Optimizador: AdamW con weight decay (1e-4)

Callbacks:

- EarlyStopping (paciencia=10)
- ReduceLROnPlateau (factor=0.5)
- ModelCheckpoint (guardar mejor modelo)

Métricas:

- Accuracy (principal)
- Pérdida (cross-entropy)
- Tiempo por época
- Uso de memoria GPU

IV. Resultados y Análisis

A. Análisis general de resultados

Durante la evaluación del desempeño de los modelos sobre el conjunto de validación, se compararon cuatro arquitecturas distintas: un modelo propio personalizado, y tres modelos basados en arquitecturas de redes neuronales profundas conocidas (VGG16, SqueezeNet y MobileNetV2). Cada modelo fue entrenado y evaluado bajo las mismas condiciones de recursos computacionales para asegurar la equidad comparativa.

El modelo propio alcanzó una precisión del 84.98%, demostrando un rendimiento competitivo con un diseño más liviano. VGG16 logró un 85.24% de precisión, lo que demuestra la efectividad del transfer learning incluso con parámetros congelados. Por su parte, SqueezeNet y MobileNetV2 mostraron los mejores resultados, alcanzando precisiones de 91.16% y 91.67% respectivamente, lo cual resalta su eficiencia tanto en precisión como en velocidad de entrenamiento.

Estos resultados indican que las arquitecturas optimizadas y preentrenadas pueden ofrecer ventajas considerables frente a modelos desarrollados desde cero, especialmente en tareas de clasificación con conjuntos de datos complejos.

B. Comparación de resultados

Puntos clave de los resultados:

- **MobileNetV2 fue el modelo con mejor desempeño**, alcanzando un **91.67% de precisión** en validación, con tiempos de entrenamiento bajos y buena capacidad de generalización.
- **SqueezeNet obtuvo un rendimiento muy cercano**, con un 91.16% de precisión, destacando por su rapidez y arquitectura compacta ideal para dispositivos con recursos limitados.
- **El modelo propio se desempeñó bien** con una precisión del 84.98%, siendo una alternativa válida cuando se requiere flexibilidad en la arquitectura sin depender de modelos preentrenados.
- **VGG16, pese a su profundidad y uso de *transfer learning***, solo superó ligeramente al modelo propio en precisión (85.24%), pero con un mayor costo computacional y tiempo de entrenamiento.

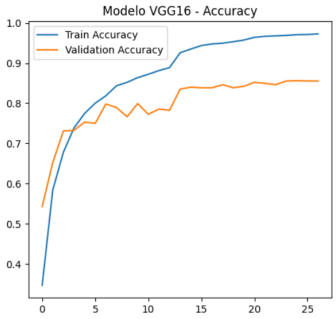

Fig. 1: Precisión (Accuracy) durante el entrenamiento para VGG16.

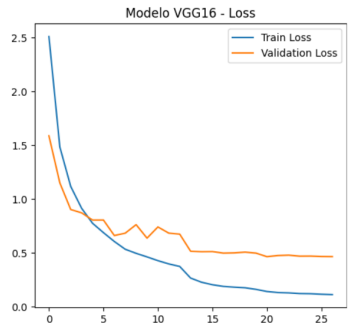

Fig. 2: Pérdida (Loss) durante el entrenamiento para VGG16.

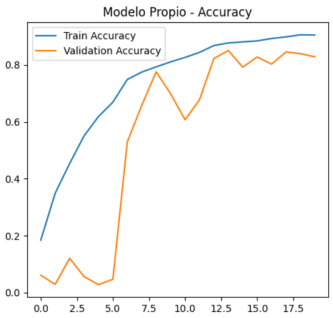

Fig. 3: Precisión (Accuracy) durante el entrenamiento para el Modelo Propio(”PapaNet”).

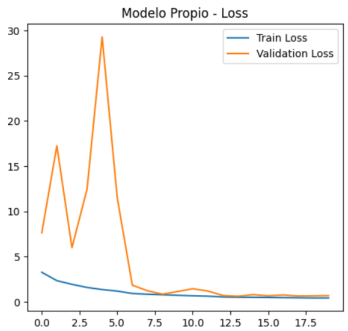

Fig. 4: Pérdida (Loss) durante el entrenamiento para el Modelo Propio(”PapaNet”).

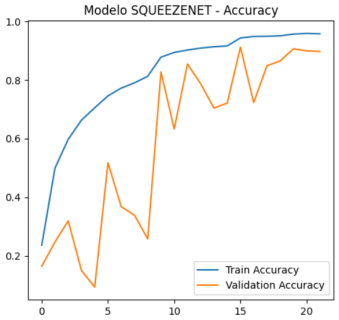

Fig. 5: Precisión (Accuracy) durante el entrenamiento para SqueezeNet.

Fig. 6: Pérdida (Loss) durante el entrenamiento para SqueezeNet.

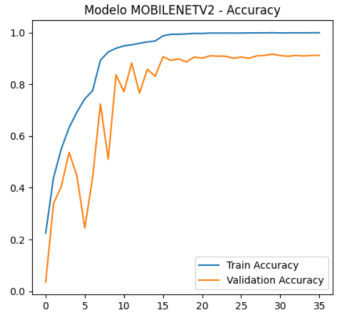

Fig. 7: Precisión (Accuracy) durante el entrenamiento para MobileNetV2.

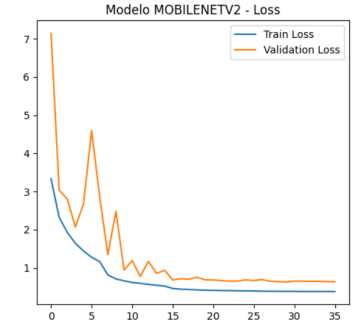

Fig. 8: Pérdida (Loss) durante el entrenamiento para MobileNetV2.

V. CONCLUSIONES

En este estudio, se evaluaron y compararon diversas arquitecturas de redes neuronales convolucionales (*CNN*) para la clasificación de 83 variedades de papas nativas peruanas. Los resultados indicaron que *MobileNetV2* alcanzó el mayor índice de precisión (*accuracy*) con un 91.67%), superando ligeramente a los modelos personalizados, así como a *VGG16* y *SqueezeNet*. La incorporación de técnicas de regularización, como *Dropout* y *Batch Normalization*, junto con la extensión de datos mediante *data augmentation*, fue esencial para mejorar la capacidad de generalización y la robustez de los modelos entrenados. Además, la implementación de *callbacks* como *EarlyStopping*, que detiene el entrenamiento si la precisión de validación no mejora en 6 épocas, y *ReduceLROnPlateau*, que ajusta de forma óptima la tasa de aprendizaje, contribuyó a evitar el sobreajuste y mejorar el rendimiento general. Como futura investigación, se sugiere explorar arquitecturas más recientes y utilizar imágenes de mayor resolución para incrementar la precisión del sistema, considerando que esto requerirá mayores recursos computacionales y entornos de ejecución más sofisticados.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 770–778, 2016.

[4] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[5] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, pp. 6105–6114, 2019.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, pp. 448–456, 2015.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Int. Conf. Learn. Represent.*, 2019.

[9] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[10] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org

[11] M. A. Pachari Lipa, B. E. Sullcarani Diaz, and S. V. Torreblanca Paz, "Repositorio de imágenes de variedades de papas nativas peruanas," GitHub, 2024. [Online]. Available: https://github.com/aprendizajeautomatico-entrega2/2DA$_E NTREGA_A PRENDIZAJE_A UTOMATICO$