

UNIVERSITÀ DEGLI STUDI DI SALERNO



DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

FONDAMENTI DI DATA SCIENCE E MACHINE LEARNING

Egyptian Hieroglyphs Models Analysis

Visual analysis for identification and classification of
hieroglyphs

Candidati:

ANTONIO GAROFALO

FULVIO SERAO

ALESSIA TURE

Professori:

PROF. GIUSEPPE POLESE

PROF. STEFANO CIRILLO

ANNO ACCADEMICO 2023/2024

Indice

1	Introduzione	4
1.1	L’idea Progettuale	4
1.2	Datasets	5
1.2.1	GlyphDataset	5
1.2.2	Altro dataset personale	6
1.3	Strumenti Utilizzati	7
1.3.1	Stato dell’arte	7
1.3.2	Tecniche di Machine Learning	9
2	Dataset	10
2.1	Architettura del Dataset	10
2.1.1	Distribuzione delle Classi	12
2.1.2	Distribuzione delle Classi Rare	13
2.1.3	Intensità dei Pixel	14
2.1.4	Distribuzione della Dimensione delle Immagini	15
2.2	Problemi Riscontrati	16
2.2.1	Eccessivo numero di classi rare	16
2.2.2	Mancanza di classi	16
2.2.3	Presenza eccessiva di rumore nelle immagini	17
2.2.4	Dataset sbilanciato	18
2.2.5	Grandezza delle immagini	18
2.3	Data Augmentation	19
2.3.1	Metodologie Utilizzate	19
2.3.2	Risultato Migliore	21

2.4	Ulteriore Dataset	23
2.4.1	Processo di estrazione	24
2.4.2	Architettura del Dataset	28
2.4.3	Distribuzione delle classi	28
2.4.4	Distribuzione della dimensione delle immagini	29
2.4.5	Ulteriori informazioni	29
3	Modelli Analizzati	30
3.1	GlyphNet (CNN)	31
3.1.1	Cos’è GlyphNet?	31
3.1.2	Come è fatto GlyphNet?	31
3.2	ATCNet (CNN)	34
3.2.1	Cos’è ATCNet?	34
3.2.2	Come è fatto ATCNet?	34
3.3	TResNet (CNN)	37
3.3.1	Cos’è TResNet?	37
3.3.2	Come è fatto TResNet?	37
3.4	Ensamble Learning	40
3.4.1	Cos’è l’Ensamble Learning?	40
3.4.2	Come è stato fatto l’Ensamble Learning?	40
3.5	Altri Stumenti: YOLO	42
3.5.1	Cos’è YOLOv5?	42
3.5.2	Come è fatto YOLOv5?	42
3.5.3	Come è stato utilizzato YOLOv5?	43
4	Analisi dei Risultati Ottenuti	45
4.1	Metriche di Valutazione	45
4.1.1	CodeCarbon	47
4.2	Metodi di Explainability	48
4.2.1	GlyphNet	50
4.2.2	ATCnet	55
4.2.3	TResNet	60

4.2.4	Ensamble Learning	64
4.2.5	YOLOv5	71
5	Conclusioni	79
5.1	Risultati a Confronto	80
5.1.1	Metriche di Performance	80
5.1.2	Metriche Energetiche	81
5.1.3	Metriche Temporali	82
5.2	Risultato migliore	82

Capitolo 1

Introduzione

Il patrimonio culturale (*Cultural Heritage*) comprende un ampio ventaglio di elementi significativi come monumenti storici, opere d’arte e aspetti culturali e storici di rilevante valore. Nell’ambito della preservazione e dell’analisi di tali beni, le tecniche di *Machine Learning (ML)* e *Natural Language Processing (NLP)* possono offrire un contributo sostanziale. La nostra idea progettuale iniziale si è concentrata sulla creazione di un sistema automatizzato per il riconoscimento e la classificazione di geroglifici, una componente cruciale per la comprensione e la preservazione della scrittura antica.

1.1 L’idea Progettuale

Il progetto è nato con l’intento di sviluppare un metodo innovativo per la classificazione automatica dei geroglifici utilizzando tecniche avanzate di Machine Learning. La nostra ipotesi era quella di utilizzare un dataset di geroglifici e applicare algoritmi di riconoscimento per facilitare la catalogazione e lo studio di questi simboli antichi, oltre ciò, definire delle didascalie intellegibili e di facile comprensione. Tale approccio avrebbe potuto contribuire significativamente alla comprensione e alla preservazione dei testi storici.

1.2 Datasets

1.2.1 GlyphDataset

Il dataset preso in considerazione per questo progetto è **GlyphDataset**, questo dataset è stato creato da Morris Franken come supporto al suo articolo ”*Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts*”[1]” presentato all’ACM Conference on Multimedia nel 2013. Il dataset è pensato per il riconoscimento automatico dei geroglifici egizi e contiene immagini di geroglifici estratti dal libro ”The Pyramid of Unas 1.1” di *Alexandre Piankoff* del 1955.

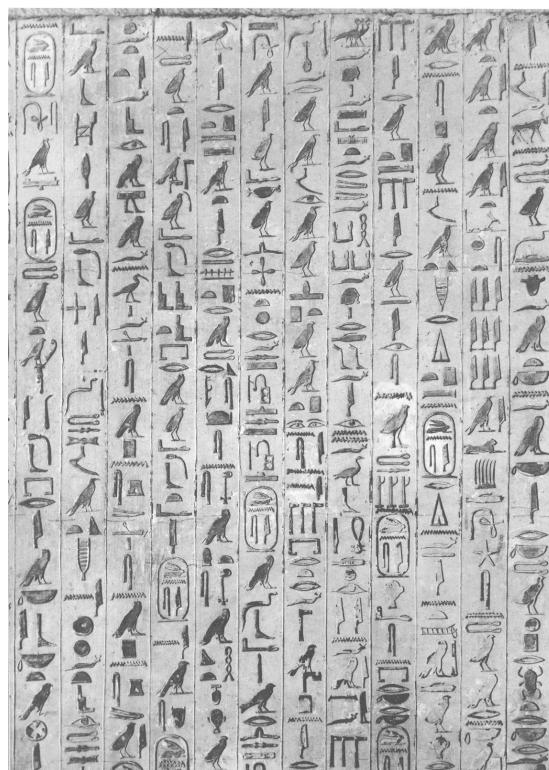


Figura 1.1: Estratto della piramide di Unas, 1955

Il dataset contiene 4210 immagini, diviso in 171 classi. Le etichette dei dati seguono lo standard ”Gardiner’s Sign List”[2], l’elenco dei segni di Gardiner è un elenco di geroglifici egizi comuni compilato da *Sir Alan Gardiner*. È considerato un riferimento standard nello studio degli antichi geroglifici egizi.

1.2.2 Altro dataset personale

A fronte di alcune problematiche riscontrate durante l'utilizzo del GlyphDataset è stato deciso di costruire un dataset personalizzato. Questo nuovo dataset è stato creato utilizzando i dati di addestramento di You Only Look Once (YOLO): poiché il suo modulo iniziale si occupa della recognition, si è pensato di utilizzarlo per riconoscere regioni di interesse (bounding boxes, utilizzate per creare nuovi campioni d'addestramento), ed etichettandoli utilizzando la label di YOLO stesso. Il risultato consiste, quindi, in un nuovo dataset, formato di immagini di alta qualità e fedeltà.

I dati sono stati reperti da COTA_COCO_anks sul sito di condivisione dati per Data Science e Machine Learning [roboflow.com](https://www.roboflow.com/).



a) Immagine originale



b) Immagine estratta

Figura 1.2: Immagine Composta (a) e Immagine singola estratta (b)

1.3 Strumenti Utilizzati

Per raggiungere l'obiettivo prefissatoci si è iniziati con una ricerca di metodologie studiate allo **stato dell'arte** riscontrando supporto da due articoli che non solo utilizzavano il dataset Glyph ma applicavano anche metodologie attute a raggiungere uno scopo simile al nostro come metodi di segmentazione e classificazione/riconoscimento dei geroglifici mediante CNN.

Gli articoli studiati sono stati "*A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification*[3]" e "*Egyptian Hieroglyphs Segmentation with Convolutional Neural Networks*[4]"

1.3.1 Stato dell'arte

L'articolo "*A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification*[3]" descrive un approccio basato sull'intelligenza artificiale, in particolare sull'Deep Learning (DL), per la classificazione dei geroglifici egiziani antichi. Gli autori esplorano l'utilizzo di Convolutional Neural Network (CNN) per classificare immagini di geroglifici provenienti da due diversi dataset.

Contesto e Motivazione

L'Artificial Intelligence (AI) e il Machine Learning (ML) stanno rivoluzionando molti campi della scienza, e recentemente hanno iniziato a influenzare anche aree come l'archeologia e la filologia. Tuttavia, mancano ancora strumenti efficaci per supportare in modo semi-automatico la decifrazione dei testi egiziani antichi. I geroglifici sono rappresentati da ideogrammi complessi, utilizzati per esprimere parole e suoni, e vengono scritti in modi diversi su vari supporti. L'automazione di questo processo di riconoscimento potrebbe semplificare la traduzione e la comprensione di questi antichi testi.

Metodologia

Gli autori hanno utilizzato tre note architetture di CNN: **ResNet-50**, **Inception-v3** e **Xception**, per addestrare e testare la classificazione di immagini di geroglifici. Hanno anche sviluppato una nuova rete, chiamata **GlyphNet**, progettata specificamente per affrontare questo compito. Le reti sono state addestrate utilizzando il paradigma del "Transfer Learning", nonché addestrate da zero, e sono stati condotti test di confronto per valutare le loro prestazioni.

Dataset

Sono stati utilizzati due dataset di immagini: uno disponibile pubblicamente (*GlyphDataset*) e l'altro creato dagli autori. Le immagini sono state pre-processate per uniformarne le dimensioni e il colore. I dataset sono stati poi combinati per creare un dataset unico, più bilanciato e adatto all'addestramento delle Neural Network.

Risultati

Gli esperimenti hanno dimostrato che GlyphNet ha superato le altre architetture in termini di accuratezza, precisione e robustezza nella classificazione dei geroglifici. La rete proposta ha mostrato migliori prestazioni anche in termini di tempo di addestramento e predizione, grazie alla sua struttura più semplice e meno soggetta a problemi di overfitting.

Conclusioni

L'articolo conclude che l'approccio basato sul Deep Learning (DL) è altamente efficace per la classificazione dei geroglifici egiziani, apre la strada a futuri sviluppi che potrebbero includere la traduzione automatica e l'analisi linguistica dei testi antichi. GlyphNet, in particolare, rappresenta un punto di partenza promettente per applicazioni più complesse nel campo dell'egittologia.

1.3.2 Tecniche di Machine Learning

Con tali conclusioni abbiamo quindi raccolto varie tecniche di Machine Learning (ML) per applicare tali studi e confrontare i risultati ottenuti da vari test per così proporre la soluzione migliore applicabile all'obiettivo prefissato. Abbiamo quindi utilizzato:

- **GlyphNet[5] (CNN)** - Modello di Convolutional Neural Network (CNN) specializzato nel riconoscimento e interpretazione di simboli complessi, come geroglifici o caratteri antichi.
- **ATCnet[6] (CNN)** - Una Neural Network che combina meccanismi di attenzione con strutture CNN per *Ancient Texts Classification*.
- **TResNet[7] (CNN)** - Convolutional Neural Network (CNN) potente e altamente efficiente, progettata per compiti di classificazione delle immagini su larga scala, che bilancia accuratezza e prestazioni computazionali.
- **Ensamble Learning** - Tecnica di Machine Learning che combina le previsioni di diversi modelli per migliorare la robustezza e l'accuratezza del risultato finale.

Un ulteriore strumento utilizzato è YOLO (You Only Look Once)[8], un algoritmo avanzato di rilevamento di oggetti in tempo reale, noto per la sua velocità e precisione nel riconoscimento di più oggetti all'interno di immagini o video.

Pytorch

Tutti i modelli di Machine Learning (ML) sopra menzionati sono stati sviluppati e gestiti utilizzando **PyTorch[9]**, una delle librerie più avanzate e versatili per il Deep Learning (DL). PyTorch è stato utilizzato come strumento principale per il training, la personalizzazione e l'ottimizzazione dei modelli, offrendo un ambiente flessibile e dinamico per lavorare con Neural Network.

Capitolo 2

Dataset

In questo capitolo descriveremo nel dettaglio il dataset utilizzato (GlyphDataset) partendo dalla sua versione base non bilanciata, per poi descrivere il processo di bilanciamento ed estrazione finale.

2.1 Architettura del Dataset

L'architettura del dataset utilizzato è così composta:

- **N° Classi** - 171
- **N° Immagini** - 4210
- **Dimensione Immagini** - 50x75
- **Numero di canali** - 1 (scala di grigi)

Le immagini sono state classificate seguendo il ”*Gardiner's Sign List*”^[2] presentando anche una classe ”**UNKNOWN**” nella quale sono catalogate 179 immagini di geroglifici non ancora riconosciuti.

Un estratto delle immagini è presente nella seguente figura 2.1:



Figura 2.1: Campioni del dataset Glyph.

Analizzando il dataset sono poi state raccolte ulteriori informazioni e statistiche cruciali per una attenta valutazione della qualità:

- **Distribuzione delle Classi** - esame della distribuzione delle classi per individuare eventuali sbilanciamenti o distorsioni nei dati.
- **Distribuzione delle Classi Rare** - identificazione e valutazione delle classi con bassa frequenza all'interno del dataset.
- **Intensità dei Pixel** - analisi della distribuzione dell'intensità dei pixel per valutare la qualità delle immagini, la presenza di rumore o altre anomalie.
- **Distribuzione della Dimensione delle Immagini** - analisi delle dimensioni (larghezza e altezza) delle immagini presenti nel dataset, per identificare la varietà e la presenza di eventuali anomalie dimensionali.

2.1.1 Distribuzione delle Classi

Le classi come anticipato sono 171 che ricoprono un totale del 23.3% di tutte le classi presenti del Gardiner's Sign List che comprende 731 caratteri catalogati in 26 gruppi.

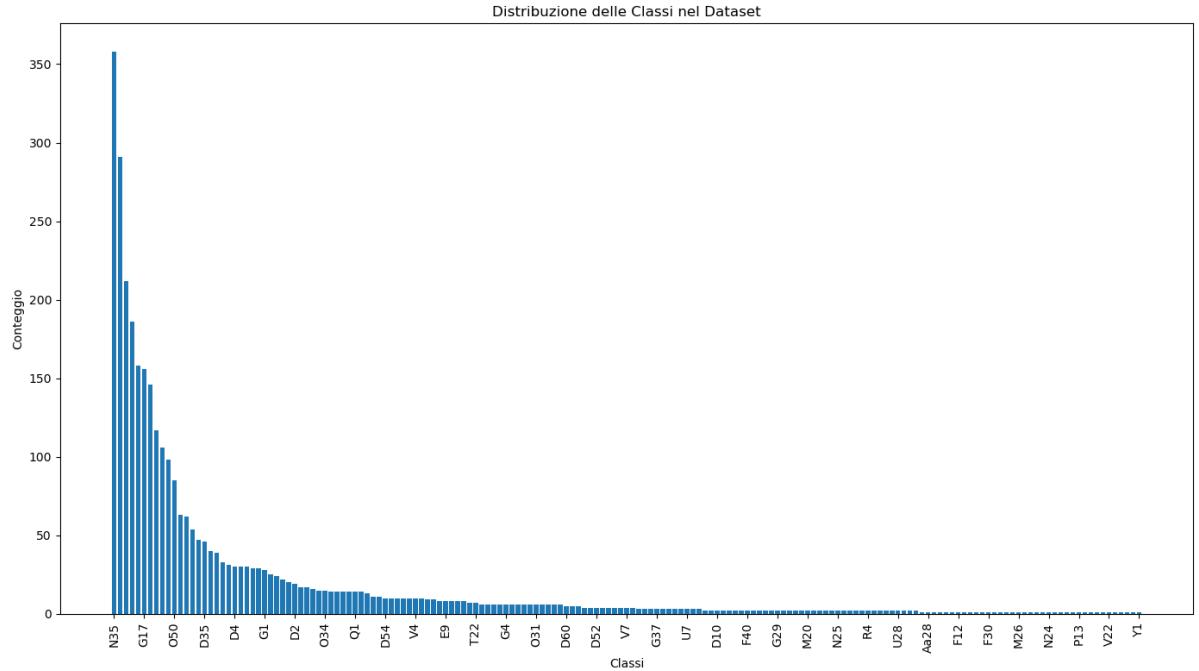


Figura 2.2: Istogramma della distribuzione delle classi.

Notiamo dalla figura 2.2 che il numero dei campioni nelle classi è distribuito in modo non uniforme, in particolare possiamo notare la classe "N35" possiede ben 358 campioni rispetto alle altre classi che si aggirano su una media di 75 campioni.

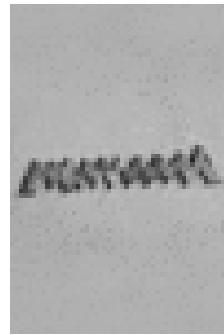


Figura 2.3: Immagine N35, "Water ripple".

2.1.2 Distribuzione delle Classi Rare

Le classi rare, ovvero quelle con pochi campioni rispetto ad altre classi, possono essere difficili da apprendere per un modello. Se non vengono adeguatamente analizzate e gestite, il modello può tendere a trascurare o classificare erroneamente queste classi, portando a una scarsa performance generale. Nel dataset Glyph abbiamo notato la seguente distribuzione delle classi rare 2.4:

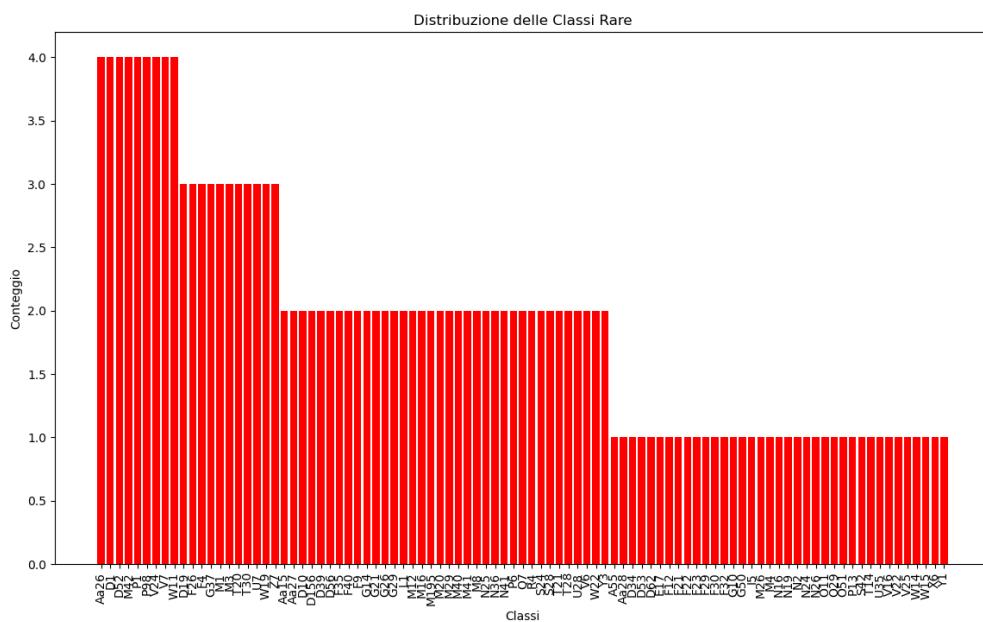


Figura 2.4: Classi Rare del dataset Glyph.

Le classi con meno di 10 campioni sono 93 ovvero più del 50% delle classi presenti. Ovviamente questo è un problema dato che il modello non avrà abbastanza campioni per generalizzare quella determinata classe.

2.1.3 Intensità dei Pixel

L'analisi dell'intensità dei pixel fornisce informazioni cruciali sulla distribuzione dei valori di luminosità all'interno delle immagini del dataset. Comprendere come sono distribuiti questi valori è fondamentale per valutare la qualità visiva e la rappresentatività del dataset, soprattutto se si considera l'importanza di preservare dettagli chiave durante la fase di pre-processing delle immagini.

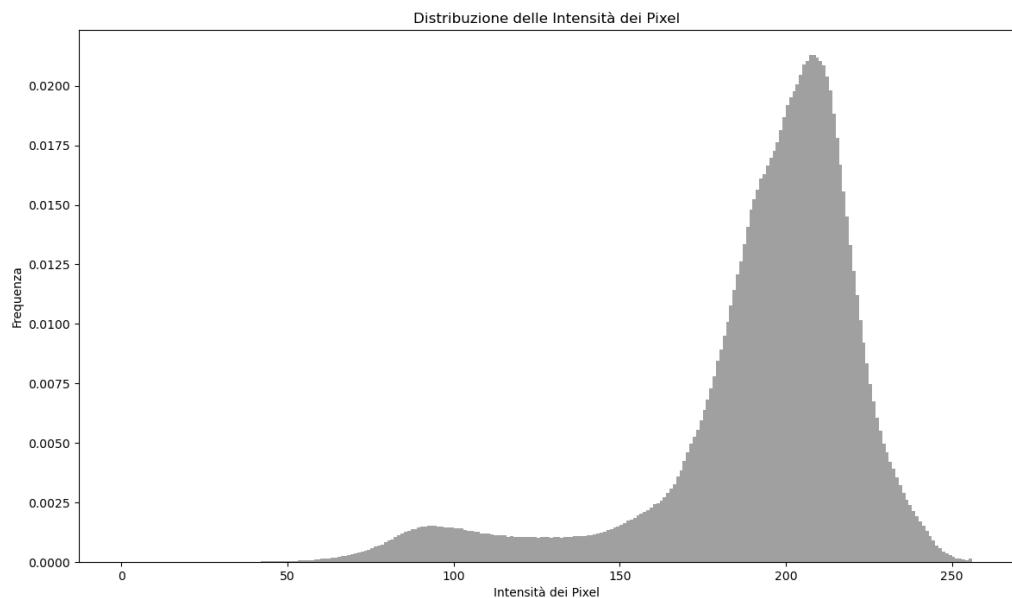


Figura 2.5: Istogramma dell'intensità dei pixel.

La distribuzione in figura 2.8 è chiaramente asimmetrica, con un picco pronunciato tra le intensità di pixel intorno ai valori alti, vicino a 200-250. Questo indica che la maggior parte dei pixel nell'immagine ha un'intensità elevata, suggerendo che l'immagine è probabilmente molto chiara o contiene molte aree di luce.

2.1.4 Distribuzione della Dimensione delle Immagini

L'analisi delle dimensioni delle immagini nel dataset è cruciale per comprendere la variabilità delle risoluzioni e per determinare eventuali necessità di pre-processing. Una distribuzione equilibrata delle dimensioni può facilitare l'addestramento del modello, mentre una variazione eccessiva potrebbe richiedere interventi specifici per garantire l'uniformità dei dati in input.

Nel dataset analizzato, abbiamo esaminato la distribuzione delle dimensioni delle immagini in termini di larghezza e altezza (in pixel). La Figura 2.6 mostra la distribuzione delle dimensioni delle immagini:

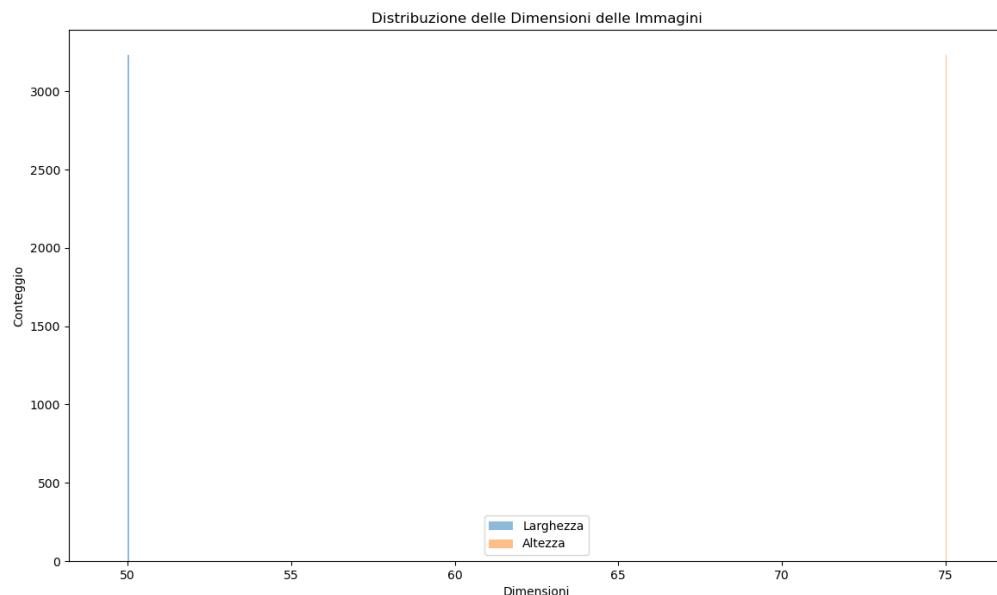


Figura 2.6: Istogramma della dimensione delle immagini.

Dall'analisi risulta che tutte le immagini hanno una distribuzione dimensionale equa pari a **50x75**.

2.2 Problemi Riscontrati

Dopo un'attenta analisi ed estrazione di varie statistiche il dataset presenta vari problemi che incidono sulle performance di analisi e valutazione dei modelli presi in considerazione. In particolare abbiamo i seguenti problemi principali:

- **Eccessivo numero di classi rare**
- **Mancanza di classi**
- **Presenza eccessiva di rumore nelle immagini**
- **Dataset sbilanciato**
- **Grandezza delle immagini**

2.2.1 Eccessivo numero di classi rare

La presenza di classi rare rappresenta una sfida perché il modello ha pochissimi dati da cui imparare per queste classi. Questo rende difficile per il modello generalizzare correttamente su queste classi, portando a errori di classificazione o una bassa capacità di riconoscere correttamente i campioni appartenenti a queste classi.

2.2.2 Mancanza di classi

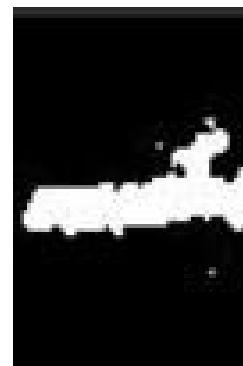
Le classi disponibili nel dataset non coprono l'intero spettro dei geroglifici studiati e catalogati nella Gardiner's Sign List. Questo implica che il modello non può fornire una didascalia accurata per i geroglifici che non sono presenti nel dataset e, pertanto, non sono stati utilizzati durante la fase di addestramento. Di conseguenza, la capacità del modello di riconoscere e classificare geroglifici non inclusi nel dataset risulta limitata. Questa lacuna può portare a una performance inadeguata nella generazione di descrizioni per i geroglifici non rappresentati, riducendo così l'affidabilità complessiva del sistema.

2.2.3 Presenza eccessiva di rumore nelle immagini

Il rumore presente nei campioni, siano essi di addestramento o di testing, ha rappresentato una problematica insormontabile, sotto molteplici punti di vista: immagini come la seguente 2.7, infatti, mostrano in maniera inconfondibile quanto trovare una soluzione di pre-processing efficace fosse una sfida ardua, e sicuramente incompatibile con tutte le immagini del set di dati.



a) Immagine non filtrata



b) Immagine filtrata

Figura 2.7: Confronto tra l'immagine originale e l'immagine filtrata.

L'immagine iniziale 2.7(a) mostra un andamento pressoché irregolare dell'intensità dei pixel, e la qualità della stessa è scadente; è stato provato un filtraggio Gaussiano per ridurre le impurità, per poi applicare il **K-Means** a due soglie. Al fine di ottenere un'immagine **0-1**, è stato utilizzato l'**Otsu Algorithm** per trovare la soglia e poi si è applicata la stessa. Il risultato finale 2.7(b), frutto dell'elaborazione appena descritta, è rappresentativo di come si siano persi tutti i dettagli più significativi dell'immagine (addirittura, il rumore presente ha creato un particolare inesistente inizialmente). Addestrare un modello con simili campioni non solo è fuorviante per lo stesso, ma risulta essere inutile, se i risultati aspettati vogliono attestarsi su soglie più che accettabili.

2.2.4 Dataset sbilanciato

In un dataset sbilanciato, il modello di apprendimento automatico tende a sviluppare una forte predisposizione verso la classe più rappresentata, con il rischio di compromettere significativamente la performance sulle classi meno rappresentate. Questo fenomeno avviene perché il modello può facilmente massimizzare la sua accuratezza complessiva semplicemente "ignorando" le classi minori, ovvero quelle con meno esempi, e concentrandosi sulla corretta classificazione della classe dominante. Di conseguenza, nonostante l'apparente buon rendimento globale, il modello può dimostrare una bassa capacità di generalizzazione, risultando inefficace nel riconoscere correttamente le istanze delle classi minoritarie.

2.2.5 Grandezza delle immagini

La dimensione delle immagini gioca un ruolo cruciale nella qualità del riconoscimento delle stesse e nell'efficacia dell'applicazione di eventuali filtri di post-elaborazione. Quando la qualità è bassa, come nel caso di una risoluzione di **50x75 pixel**, la distribuzione dei pixel risulta significativamente ridotta, limitando la quantità di dettagli che possono essere catturati. Questo compromette la precisione degli algoritmi di riconoscimento, poiché dispongono di meno informazioni visive su cui basarsi. Inoltre, l'applicazione di filtri diventa meno efficace, poiché la ridotta densità di pixel rende difficile l'identificazione e l'enfatizzazione di particolari caratteristiche dell'immagine, come bordi, texture e contrasto.

2.3 Data Augmentation

La **Data Augmentation** è una tecnica utilizzata in Machine Learning e, in particolare, nell’elaborazione di immagini per aumentare la quantità e la diversità dei dati di addestramento senza dover raccogliere nuovi dati. A fronte dei problemi riscontrati del dataset, per migliorarne la qualità abbiamo testato varie tecniche di Augmentation come ad esempio **Synthetic Minority Over-sampling Technique (SMOTE)** e **Geometric Transformations**.

2.3.1 Metodologie Utilizzate

SMOTE

La prima tecnica che si voleva utilizzare è stata **SMOTE**, essa è una tecnica utilizzata per affrontare il problema dello sbilanciamento delle classi nei dataset, specialmente in scenari di classificazione.

Come funziona SMOTE? SMOTE genera nuovi esempi sintetici per le classi minoritarie creando combinazioni lineari tra gli esempi esistenti. Invece di semplicemente duplicare gli esempi della classe minoritaria, SMOTE seleziona casualmente coppie di esempi simili e genera nuovi esempi che si trovano tra questi due punti nello spazio delle caratteristiche. Questo aumenta la rappresentanza della classe minoritaria nel dataset senza introdurre duplicati esatti, migliorando così la capacità del modello di imparare a riconoscere anche queste classi meno rappresentate.

Limiti di SMOTE L’implementazione di o più esempi per ciascuna classe per poter generare nuovi dati sintetici. Quando una classe ha un solo esempio, SMOTE non può essere applicato, perché non ci sono altri punti simili con cui creare combinazioni. Questo rappresenta un problema particolare nei dataset con classi estremamente sbilanciate, dove alcune classi possono avere solo uno o pochissimi esempi.

Geometric Transformations

Un'altra tecnica di Data Augmentation che abbiamo utilizzato è rappresentata dalle **Geometric Transformations**. Questa tecnica include varie trasformazioni geometriche applicate alle immagini esistenti per creare nuove varianti dei dati di addestramento, aumentando così la diversità del dataset.

Come funzionano le Geometric Transformations? Le trasformazioni geometriche modificano l'orientamento e la struttura delle immagini originali attraverso operazioni come:

- **Flip Orizzontale e Verticale:** Capovolgere l'immagine rispettivamente lungo l'asse orizzontale o verticale, creando versioni speculari dell'immagine.
- **Rotazione:** Ruotare l'immagine di un certo angolo per variare l'orientamento dei contenuti.
- **Ridimensionamento Casuale:** Le immagini vengono ridimensionate casualmente, con un fattore di scala variabile. Questo altera leggermente le dimensioni dell'immagine, aiutando il modello a imparare a riconoscere i simboli indipendentemente dalle dimensioni

Vantaggi delle Geometric Transformations Queste trasformazioni aiutano a rendere il modello meno sensibile alle variazioni di posizione, orientamento e scala delle immagini, migliorando la capacità di generalizzare su dati non visti. Inoltre, le Geometric Transformations non richiedono la presenza di più esempi per ogni classe, rendendole particolarmente utili in situazioni in cui alcune classi hanno un numero limitato di esempi.

2.3.2 Risultato Migliore

Dopo aver testato varie tecniche, le Geometric Transformations si sono rivelate particolarmente efficaci nel migliorare la qualità del dataset, soprattutto per le classi con pochi esempi. In particolare, l'uso di rotazioni, flip, e traslazioni ha contribuito a creare un dataset più diversificato e bilanciato, migliorando le prestazioni del modello sui dati di test.

Ecco come è stata fatta l'augmentation seguendo le linee guida dell'articolo ”*A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification*[3]”:

- **Flip Verticale Casuale** - Le immagini vengono capovolte verticalmente con una certa probabilità. Questo introduce varianti delle immagini che potrebbero essere ribaltate lungo l'asse verticale, aumentando la diversità del set di dati.
- **Flip Orizzontale Casuale** - Le immagini vengono capovolte orizzontalmente con una certa probabilità, creando versioni speculari delle immagini originali.
- **Rotazione Casuale** - Le immagini vengono ruotate di un angolo casuale scelto tra -10 e +10 gradi. Questa trasformazione aiuta a rendere il modello meno sensibile alla posizione esatta dei simboli nelle immagini.
- **Ridimensionamento Casuale** - Le immagini vengono ridimensionate casualmente, con un fattore di scala variabile tra 0.95 e 1.05. Questo altera leggermente le dimensioni dell'immagine, aiutando il modello a imparare a riconoscere i simboli indipendentemente dalle dimensioni.
- **Traslazione Casuale** - Le immagini vengono traslate orizzontalmente e verticalmente fino al 10% delle dimensioni dell'immagine. Questo sposta il contenuto dell'immagine in modo casuale, simulando piccole variazioni di posizionamento.

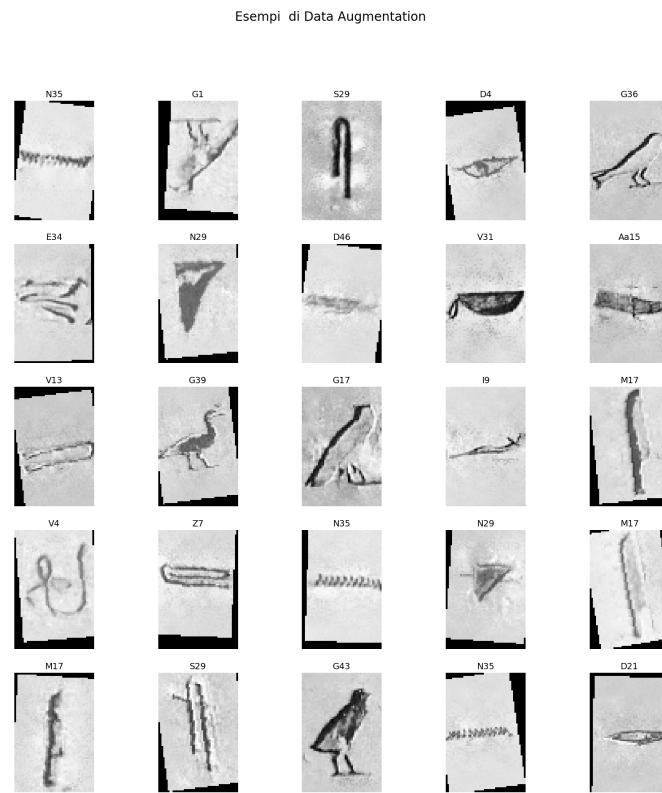


Figura 2.8: Esempio di Data Augmentation.

Ogni immagine nel set di training viene sottoposta a queste trasformazioni, creando un totale di cinque immagini aumentate per ogni immagine originale.

2.4 Ulteriore Dataset

Come accennato nell'introduzione, un altro dataset utilizzato è stato ricavato dal set di dati utilizzato da YOLO. Il numero di classi è stato ridotto a 50 (dalle iniziali 171), ed è stata modificata la nomenclatura delle classi (trasformandola da semplici identificativi alfanumerici a parole autodefinite, come ad esempio "uomo"). La procedura ha consentito di tirare fuori numerose immagini di alta qualità, che hanno permesso di creare un dataset di discreto livello per future classificazioni.



Figura 2.9: Campioni del dataset estratto.

2.4.1 Processo di estrazione

La funzione ‘*convert_yolo_to_classification*’ è progettata per trasformare un dataset creato per il rilevamento di oggetti con You Only Look Once (YOLO) in un dataset adatto alla classificazione delle immagini. Questo processo implica la conversione delle annotazioni delle **bounding boxes** (che identificano la posizione degli oggetti nelle immagini) in singole immagini etichettate, ciascuna appartenente a una classe specifica.

Come avviene la conversione

1. **Conversione delle Coordinate:** Le annotazioni You Only Look Once (YOLO) forniscono le coordinate delle **bounding boxes** in un formato normalizzato, dove i valori vanno da 0 a 1, indicando le proporzioni dell’immagine piuttosto che le posizioni esatte in pixel. La prima operazione eseguita dalla funzione è quindi la conversione di queste coordinate normalizzate in coordinate di pixel reali, calcolate in base alla larghezza e all’altezza effettiva dell’immagine. Questo passaggio è cruciale per poter poi ritagliare l’immagine correttamente.
2. **Determinazione della Bounding Box:** Una volta ottenute le coordinate in pixel, si determina la bounding box minima che racchiude tutti i punti relativi all’oggetto annotato. Questa bounding box rappresenta l’area dell’immagine che verrà ritagliata per isolare l’oggetto.
3. **Processamento del Dataset:** La funzione opera su tre parti del dataset: *train*, *valid*, e *test*, che corrispondono ai set di addestramento, validazione, e test. Per ciascuna di queste parti, la funzione scorre attraverso tutte le immagini e i relativi file di annotazione, applicando le trasformazioni descritte.
4. **Ritaglio delle Immagini:** Utilizzando le bounding boxes calcolate, la funzione ritaglia l’immagine originale per ottenere un’immagine più piccola che contiene solo l’oggetto di interesse. Questo ritaglio è fondamentale

per trasformare il problema di rilevamento degli oggetti in un problema di classificazione, dove ogni immagine rappresenta una singola istanza di una classe.

5. **Salvataggio delle Immagini:** Dopo il ritaglio, le immagini vengono salvate in una struttura di directory organizzata per classi. Ogni classe ha la sua cartella, e le immagini ritagliate vengono salvate al suo interno con nomi che includono informazioni sulla loro posizione originale, per evitare conflitti di nomi. Questo passaggio organizza il dataset in modo che possa essere facilmente utilizzato per addestrare modelli di classificazione.

Python Code

Ecco il codice sviluppato per l'estrazione del dataset:

```
1 def convert_yolo_to_classification(dataset_dir='dataset', output_dir='classification_dataset'):
2     # Funzione per convertire coordinate normalizzate in coordinate di pixel
3     def normalize_to_pixel_coords(coords, img_width, img_height):
4         pixel_coords = []
5         for i in range(0, len(coords), 2):
6             x = int(coords[i] * img_width)
7             y = int(coords[i + 1] * img_height)
8             pixel_coords.append((x, y))
9
10        return pixel_coords
11
12    # Funzione per ottenere la bounding box minima da una serie di punti
13    def get_bounding_box_from_points(points):
14        x_min = min(point[0] for point in points)
15        y_min = min(point[1] for point in points)
16        x_max = max(point[0] for point in points)
17        y_max = max(point[1] for point in points)
18
19        return x_min, y_min, x_max, y_max
20
21    # Processare ogni parte del dataset (train, val, test)
22    for split in ['train', 'valid', 'test']:
23        split_images_dir = os.path.join(dataset_dir, split, 'images')
24        split_labels_dir = os.path.join(dataset_dir, split, 'labels')
25        split_output_dir = os.path.join(output_dir, split)
26
27        # Creare la struttura di output
28        if not os.path.exists(split_output_dir):
29            os.makedirs(split_output_dir)
30
31        # Processare ogni immagine e file di annotazione
32        for label_file in os.listdir(split_labels_dir):
33            with open(os.path.join(split_labels_dir, label_file), 'r') as f:
34                annotations = f.readlines()
35
36                image_path = os.path.join(split_images_dir, label_file.replace('.txt', '.jpg'))
37                image = cv2.imread(image_path)
38                img_height, img_width = image.shape[:2]
39
40                for annotation in annotations:
41                    values = list(map(float, annotation.split()))
42
43                    class_id = int(values[0])
44                    coords = values[1:] # Tutte le coppie di coordinate
```

```

44     if len(coords) % 2 != 0:
45         print(f"Errore: numero dispari di valori nelle coordinate: {coords}")
46         continue
47
48     # Converti le coordinate normalizzate in coordinate di pixel
49     points = normalize_to_pixel_coords(coords, img_width, img_height)
50
51     # Ottieni la bounding box che contiene tutti i punti
52     x_min, y_min, x_max, y_max = get_bounding_box_from_points(points)
53
54     # Ritagliare l'immagine
55     cropped_img = image[y_min:y_max, x_min:x_max]
56
57     # Salva l'immagine ritagliata nella cartella della classe corrispondente
58     class_folder = os.path.join(split_output_dir, str(class_id))
59     if not os.path.exists(class_folder):
60         os.makedirs(class_folder)
61
62     img_name = os.path.basename(image_path).replace('.jpg', f'{x_min}_{y_min}.jpg')
63     cv2.imwrite(os.path.join(class_folder, img_name), cropped_img)
64
65     print("Conversione completata!")

```

Listing 2.1: Funzione di estrazione

Sintesi

La funzione automatizza il processo di conversione di un dataset You Only Look Once (YOLO), che è orientato al rilevamento di oggetti, in un dataset di classificazione delle immagini. Ogni immagine originale viene suddivisa in immagini più piccole, ciascuna contenente un singolo oggetto, che viene poi classificato e salvato in una struttura di directory basata sulle classi. Questo permette di utilizzare il dataset convertito per addestrare modelli di classificazione, fornendo un metodo semplice ed efficace per riutilizzare i dati di rilevamento in un contesto diverso.

2.4.2 Architettura del Dataset

L'architettura del dataset utilizzato è così composta:

- **N° Classi** - 50
- **N° Immagini** - 108.787
- **Dimensione Immagini** - variabile
- **Numero di canali** - 3 (colori)

2.4.3 Distribuzione delle classi

La distribuzione delle classi in figura 2.10 ci dimostra come il lavoro di estrazione ha portato ad un dataset sicuramente più bilanciato del precedente avendo una media di campioni molto più alta e l'assenza di classi rare.

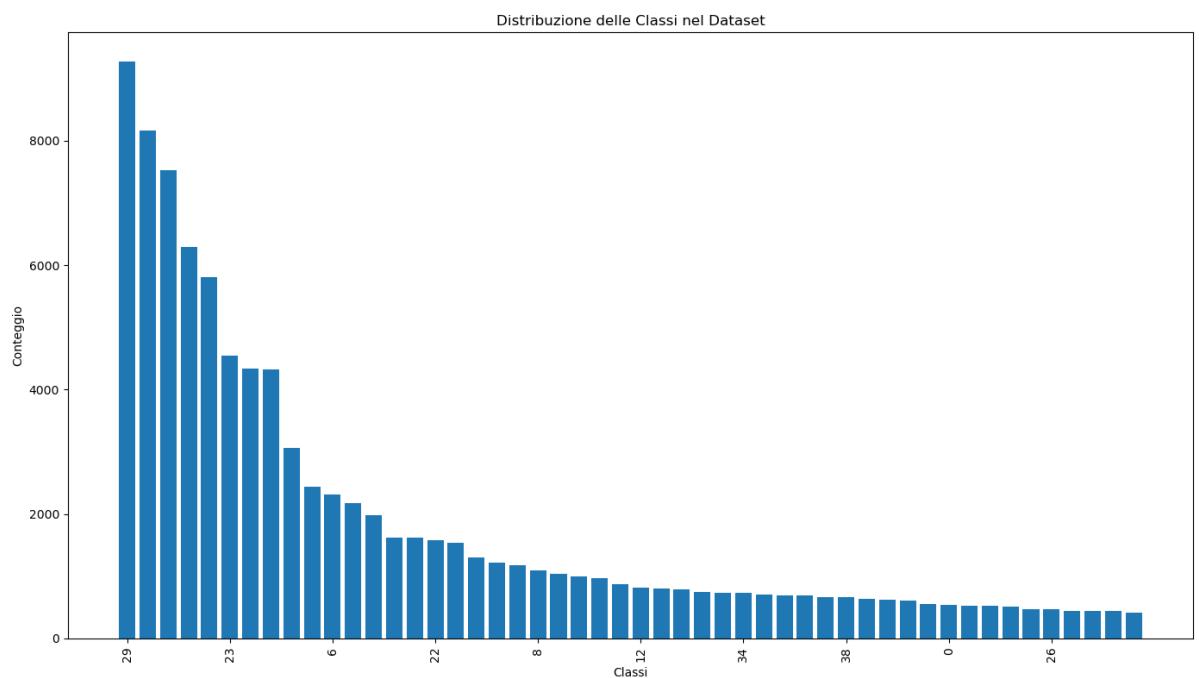


Figura 2.10: Istogramma della distribuzione delle classi.

2.4.4 Distribuzione della dimensione delle immagini

La distribuzione della dimensione delle immagini, ovviamente, dimostra che le immagini presenti sono di dimensione variabile ed eterogenea, il che seppur il dataset è di ottima qualità può essere un problema che è stato correttamente gestito attraverso un opportuno resize standard 224 pixel.

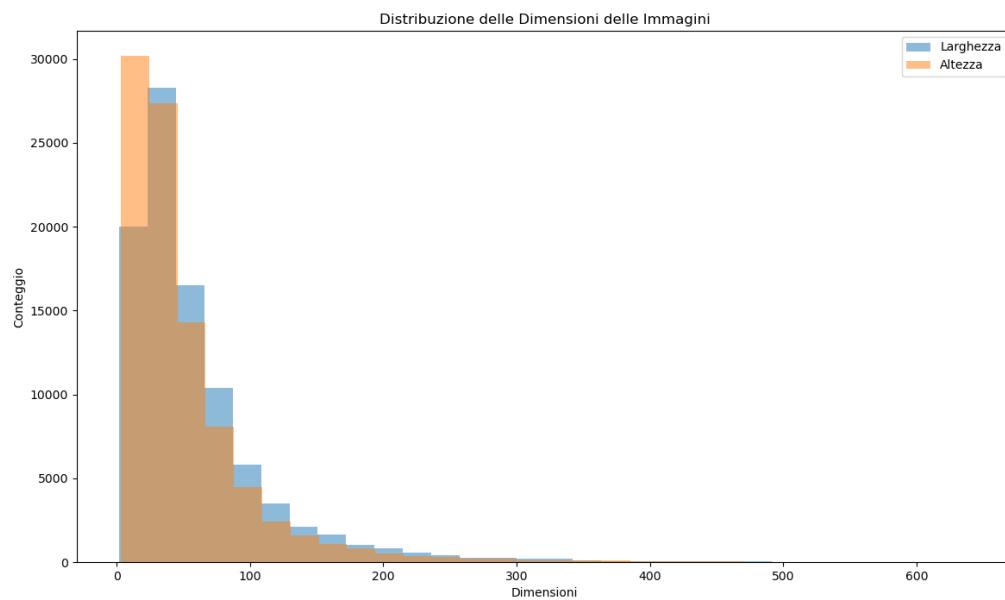


Figura 2.11: Istogramma della dimensione delle immagini.

2.4.5 Ulteriori informazioni

Nel dataset estratto secondo le metodologie descritte prima non è stato considerata l'istogramma dell'intensità dei pixel dato che le immagini si presentavano a 3 canali (a colori).

Capitolo 3

Modelli Analizzati

In virtù dei dataset utilizzati, e delle problematiche riscontrate, si è deciso di ricorrere a diversi approcci per quanto concerne il modulo di classificazione, tra cui:

- **GlyphNet (CNN);**
- **AtcNET (CNN);**
- **TResNet (CNN);**
- **Ensamble Learning.**

Si è inoltre deciso di utilizzare **You Only Look Once (YOLO)**, un modulo che abbina alla fase di classificazione anche quella di **image recognition** (che, normalmente, andrebbe fatta in un modulo a parte nelle normali CNN).

3.1 GlyphNet (CNN)

3.1.1 Cos'è GlyphNet?

GlyphNet è una Convolutional Neural Network (CNN) progettata per la classificazione di immagini, con un focus particolare sul riconoscimento di geroglifici o altri simboli complessi utilizzata nell'articolo di riferimento[3]. Il design di GlyphNet sfrutta l'architettura modulare, composta da diversi blocchi specializzati, ognuno dei quali ha un ruolo specifico nell'elaborazione delle immagini. Questa struttura a blocchi permette al modello di essere efficiente dal punto di vista computazionale, pur mantenendo un'elevata capacità di estrarre e interpretare caratteristiche visive complesse.

3.1.2 Come è fatto GlyphNet?

La rete è suddivisa in tre blocchi principali: il **First Block**, l'**Inner Block** e il **Final Block**. Ognuno di questi blocchi esegue specifiche operazioni di convoluzione e pooling che trasformano progressivamente l'immagine in input fino a produrre una classificazione finale.

1. **First Block: L'elaborazione iniziale delle immagini** Il primo blocco della rete, chiamato First Block, ha il compito di processare l'immagine grezza in input e trasformarla in una rappresentazione che può essere ulteriormente elaborata dai blocchi successivi.
 - **Convoluzioni:** Il First Block utilizza due convoluzioni standard. Le convoluzioni sono operazioni che applicano filtri all'immagine per estrarre caratteristiche locali come bordi, texture, e pattern semplici. Ogni convoluzione è seguita da una normalizzazione batch (BatchNorm), che stabilizza e accelera l'addestramento, e da un'operazione di pooling (MaxPool), che riduce la dimensione delle feature maps mantenendo le informazioni essenziali.
 - **Funzione:** L'obiettivo di questo blocco è ridurre la complessità dell'immagine in termini di dimensioni, mentre estrae le caratteristiche

visive più rilevanti. Il risultato di questo blocco è una serie di feature maps che rappresentano le caratteristiche fondamentali dell'immagine.

2. Inner Block: Raffinamento e approfondimento delle caratteristiche

Il secondo blocco, chiamato Inner Block, è progettato per estrarre caratteristiche più complesse e raffinate dall'immagine.

- **Convoluzioni Separabili:** Questo blocco utilizza convoluzioni separabili (depthwise e pointwise). La convoluzione depthwise applica filtri indipendenti a ciascun canale dell'immagine, mentre la convoluzione pointwise combina questi canali in nuove feature maps. Questo approccio riduce il numero di parametri e il carico computazionale, rendendo il modello più efficiente senza compromettere la capacità di estrazione delle caratteristiche.
- **Inspiration from Inception:** Il design di questo blocco si ispira ai blocchi Inception, che combinano operazioni convoluzionali di diversa natura per catturare diverse scale di caratteristiche. Le operazioni di convoluzione separabile seguite da pooling e normalizzazione batch aiutano a migliorare la rappresentazione delle caratteristiche.
- **Funzione:** Questo blocco prende le feature maps create dal First Block e le trasforma in rappresentazioni ancora più dettagliate e complesse, che possono rappresentare aspetti più astratti e distintivi dell'immagine, come la forma complessiva di un simbolo o combinazioni specifiche di pattern.

3. Final Block: Preparazione dell'output finale

Il blocco finale, chiamato Final Block, è responsabile della produzione della classificazione finale.

- **Separable Convolution e Pooling Globale:** Anche questo blocco utilizza convoluzioni separabili per ulteriormente raffinare le caratteristiche. Dopo questa convoluzione, viene applicato un pooling globale (global average pooling) che riduce ciascuna feature map a un singolo valore medio, sintetizzando le informazioni chiave.

- **Dropout e MLP:** Per evitare l'overfitting, un'operazione di dropout disattiva casualmente una parte dei neuroni durante l'addestramento, forzando il modello a generalizzare meglio. L'output viene quindi passato attraverso un Multi-Layer Perceptron (MLP) che produce un vettore di output con una dimensione pari al numero di classi da predire.
- **Softmax:** Infine, il risultato viene passato attraverso una funzione softmax che converte le uscite del MLP in probabilità logaritmiche, indicando la classe più probabile a cui l'immagine appartiene.

3.2 ATCNet (CNN)

3.2.1 Cos'è ATCNet?

ATCNet (**Ancient Texts Classification**) è una Convolutional Neural Network (CNN) progettata per la classificazione di immagini, con un'architettura specializzata che sfrutta depthwise separable convolutions per migliorare l'efficienza computazionale, mantenendo al contempo elevate prestazioni nella classificazione. L'architettura è composta da vari blocchi convoluzionali, ciascuno dei quali è progettato per estrarre e raffinare caratteristiche specifiche dall'immagine in input.

3.2.2 Come è fatto ATCNet?

La rete è suddivisa in vari blocchi chiave che eseguono operazioni di convoluzione, pooling e normalizzazione, trasformando l'immagine in input fino a produrre una classificazione finale.

1. **Blocchi Iniziali: Estrazione delle caratteristiche di base** I primi due blocchi della rete (conv1 e conv2) sono responsabili dell'estrazione delle caratteristiche visive di base dall'immagine.
 - **Convoluzioni Standard:** Questi blocchi utilizzano convoluzioni standard per elaborare l'immagine iniziale e iniziare a estrarre caratteristiche semplici come bordi e texture. Ogni convoluzione è seguita da una normalizzazione batch (BatchNorm) per stabilizzare l'addestramento e da un'operazione di pooling (MaxPool) che riduce la dimensione delle feature maps mentre aumenta l'invarianza a traslazioni.
 - **Funzione:** Questi blocchi iniziali trasformano l'immagine in una rappresentazione preliminare che può essere ulteriormente raffinata nei blocchi successivi.
2. **Blocchi Separable Convolution: Raffinamento e approfondimento delle caratteristiche** La parte centrale dell'architettura di ATCNet

è costituita da quattro blocchi sequenziali (sepconv1, sepconv2, sepconv3, sepconv4) che utilizzano convoluzioni separabili in profondità.

- **Convoluzioni Separabili in Profondità:** Ogni blocco separa la convoluzione in due fasi: una convoluzione depthwise, che applica filtri separati a ciascun canale dell'immagine, e una convoluzione pointwise, che combina questi canali in nuove feature maps. Questo approccio riduce significativamente il numero di parametri e il carico computazionale rispetto alle convoluzioni standard.
- **Batch Normalization e Pooling:** Ogni blocco include operazioni di batch normalization e pooling, che aiutano a mantenere la stabilità del modello durante l'addestramento e a ridurre ulteriormente la dimensionalità delle feature maps.
- **Funzione:** Questi blocchi sono progettati per catturare caratteristiche più complesse e raffinate dell'immagine, migliorando la capacità del modello di riconoscere pattern distintivi necessari per una classificazione accurata.

3. **Blocco Finale: Preparazione dell'output** Il blocco finale della rete (exit_block) è responsabile della preparazione delle caratteristiche finali per la classificazione.

- **Convoluzione Finale e Pooling Globale:** Il blocco finale applica una convoluzione separabile seguita da un'operazione di pooling globale (Global Average Pooling), che riduce ogni feature map a un singolo valore medio. Questo passaggio sintetizza le informazioni chiave estratte dall'immagine.
- **Dropout e Fully Connected Layer:** Dopo il pooling globale, viene applicato un dropout per evitare l'overfitting. L'output viene quindi passato attraverso un livello completamente connesso (Fully Connected Layer), che produce le previsioni finali della rete.

- **Funzione:** Questo blocco è essenziale per integrare tutte le caratteristiche estratte dai blocchi precedenti e produrre un output che rappresenta la probabilità che l'immagine appartenga a ciascuna delle classi di destinazione.

3.3 TResNet (CNN)

3.3.1 Cos'è TResNet?

TResNet è una Convolutional Neural Network (CNN) progettata per la classificazione di immagini su larga scala, con un focus particolare sull'ottimizzazione delle prestazioni computazionali senza compromettere l'accuratezza. TResNet è costruito per essere altamente efficiente, utilizzando tecniche avanzate di convoluzione e ottimizzazione che permettono di gestire dataset di grandi dimensioni e complessità. Questa rete è particolarmente adatta per applicazioni dove è necessario bilanciare potenza di calcolo e precisione.

3.3.2 Come è fatto TResNet?

La rete TResNet è suddivisa in vari blocchi funzionali che eseguono operazioni di convoluzione, pooling, e normalizzazione, progettati per estrarre, raffinare e classificare caratteristiche visive complesse dalle immagini in input. In questo progetto, è stato utilizzato un modello TResNet pre-addestrato su ImageNet, sfruttando il **Transfer Learning** per adattarlo al compito specifico di classificazione delle immagini di interesse.

1. **Configurazione Iniziale: Pre-processamento dei dati**
2. **Blocchi di Convoluzione: Estrazione delle caratteristiche** TResNet utilizza un'architettura di convoluzioni ottimizzate per estrarre caratteristiche a diverse scale e complessità.
 - **Convoluzioni e Batch Normalization:** Ogni blocco di convoluzione è seguito da una normalizzazione batch (BatchNorm), che stabilizza l'addestramento riducendo la variazione nei dati tra batch consecutivi, e da una funzione di attivazione ReLU, che introduce non linearità nel modello.

- **Funzione:** Questi blocchi sono fondamentali per estrarre caratteristiche di basso e alto livello dall'immagine, necessarie per distinguere tra diverse classi di oggetti.
3. **Blocchi Avanzati: Convoluzioni Separate e Attenzione** TResNet integra blocchi di convoluzione separabili in profondità e meccanismi di attenzione per migliorare l'efficienza e la capacità di focus su regioni importanti dell'immagine.
- **Convoluzioni Depthwise e Pointwise:** Queste convoluzioni separano il processo di convoluzione in due fasi: una convoluzione depthwise, che opera su ciascun canale separatamente, seguita da una convoluzione pointwise, che combina i canali per creare nuove feature maps. Questo approccio riduce il numero di parametri e la complessità computazionale.
 - **Attenzione Squeeze-and-Excitation (SE):** Alcuni blocchi includono meccanismi di attenzione come SE, che permettono alla rete di enfatizzare le caratteristiche più rilevanti delle immagini, migliorando l'accuratezza senza aggiungere eccessiva complessità.
 - **Funzione:** Questi blocchi raffinano ulteriormente le caratteristiche estratte, assicurando che il modello mantenga l'attenzione sulle parti più critiche dell'immagine per una classificazione accurata.
4. **Blocco Finale: Preparazione dell'output** Il blocco finale della rete TResNet è responsabile della produzione della classificazione finale delle immagini.
- **Global Average Pooling e Fully Connected Layer:** Dopo l'ultima serie di convoluzioni, viene applicato un pooling globale (Global Average Pooling) per ridurre ciascuna feature map a un singolo valore medio. Questo output viene poi passato attraverso un livello completamente connesso (Fully Connected Layer), che produce il vettore di output con la previsione della classe.

- **Funzione:** Questo blocco integra tutte le caratteristiche estratte dai blocchi precedenti e produce l'output finale sotto forma di probabilità per ciascuna classe, completando il processo di classificazione.

3.4 Ensamble Learning

3.4.1 Cos'è l'Ensamble Learning?

Ensamble Learning è una tecnica di Machine Learning (ML) che combina le previsioni di più modelli per migliorare l'accuratezza e la robustezza della classificazione rispetto a un singolo modello. L'idea di base è che combinando le previsioni di modelli diversi, si possono mitigare gli errori individuali, ottenendo così un modello finale più performante. L'ensemble può essere realizzato in vari modi, tra cui il **bagging**, il **boosting** e il **voting**. In questo progetto, abbiamo utilizzato un approccio basato sul **soft voting** ponderato, combinando le previsioni di più modelli TResNet pre-addestrati.

3.4.2 Come è stato fatto l'Ensamble Learning?

Il processo di Ensamble Learning è stato realizzato attraverso diverse fasi chiave, ognuna delle quali contribuisce a costruire un modello finale che bilancia accuratezza e robustezza.

1. **Addestramento dei Modelli Singoli** L'ensemble è composto da modelli TResNet addestrati individualmente, ognuno dei quali è stato ottimizzato per il compito specifico di classificazione delle immagini.
 - **Modelli Pre-addestrati:** Per ogni modello, abbiamo sfruttato un TResNet pre-addestrato su ImageNet, adattandolo attraverso il **transfer learning** per migliorare le prestazioni su un dataset specifico.
 - **Addestramento:** Ogni modello è stato addestrato indipendentemente, registrando le migliori prestazioni su un set di validazione. I pesi di ciascun modello sono stati salvati per l'uso successivo nell'ensemble.
 - **Obiettivo:** Questa fase permette di creare un insieme di modelli diversificati, ciascuno dei quali è specializzato nel riconoscere pattern specifici nel dataset.

2. **Calcolo dei Pesi dei Modelli** Dopo l'addestramento, ciascun modello è stato valutato su un set di validazione per determinare la sua accuratezza.

- **Valutazione delle Prestazioni:** L'accuratezza di ciascun modello è stata calcolata utilizzando il set di validazione, e queste accuratezze sono state utilizzate per assegnare un peso a ciascun modello.
- **Ponderazione:** I pesi assegnati riflettono la fiducia nel modello: maggiore è l'accuratezza, maggiore sarà il peso del modello nel processo di combinazione delle predizioni.
- **Funzione:** Questo processo di ponderazione assicura che i modelli con migliori prestazioni abbiano un'influenza maggiore nel modello finale.

3. **Combinazione delle Predizioni (Soft Voting)** Una volta calcolati i pesi, le predizioni dei singoli modelli sono state combinate utilizzando un metodo di **soft voting** ponderato.

- **Predizioni delle Probabilità:** Ogni modello produce una distribuzione di probabilità sulle classi possibili per ciascuna immagine. Queste probabilità vengono combinate pesandole in base alle performance dei modelli.
- **Soft Voting:** Le probabilità combinate vengono utilizzate per determinare la classe finale, scegliendo quella con la probabilità complessiva più alta. Il soft voting ponderato consente di sfruttare al meglio le capacità di ciascun modello, migliorando l'accuratezza complessiva rispetto a qualsiasi singolo modello.

3.5 Altri Stumenti: YOLO

3.5.1 Cos'è YOLOv5?

You Only Look Once (YOLO) è un modello di Deep Learning (DL) altamente efficiente e veloce, progettato per il rilevamento di oggetti in tempo reale. A differenza delle tradizionali Convolutional Neural Network (CNN) utilizzate per la classificazione di immagini, YOLOv5 è in grado di individuare e classificare più oggetti all'interno di un'immagine simultaneamente, rendendolo uno strumento ideale per applicazioni come la videosorveglianza, il monitoraggio del traffico e la ricerca visiva.

3.5.2 Come è fatto YOLOv5?

YOLOv5 è una rete neurale convoluzionale (CNN) che adotta un'architettura a più livelli progettata per il rilevamento rapido ed efficiente degli oggetti. L'architettura di YOLOv5 è composta da tre componenti principali: il **backbone**, il **neck**, e la testa (**head**).

1. Backbone

- Il backbone di YOLOv5 è una rete neurale pre-addestrata utilizzata per estrarre le caratteristiche fondamentali delle immagini in input. Questa parte dell'architettura è responsabile della conversione dell'immagine grezza in rappresentazioni intermedie, note come feature maps, che catturano gli elementi visivi chiave, come bordi, texture, e forme.
- YOLOv5 utilizza vari tipi di backbone, come CSPDarknet53, ottimizzati per bilanciare l'accuratezza e la velocità di elaborazione.

2. Neck

- Il neck è la parte dell'architettura che aggredisce le feature maps provenienti dal backbone a diverse scale. Questo processo è noto come Feature Pyramid Network (FPN) e Path Aggregation Network (PANet), che consentono al modello di considerare sia le caratteristiche a
-

bassa risoluzione (utile per il rilevamento di grandi oggetti) sia quelle ad alta risoluzione (utile per il rilevamento di piccoli oggetti).

- Questa parte dell'architettura è fondamentale per migliorare la precisione del rilevamento su oggetti di diverse dimensioni e posizioni all'interno dell'immagine.

3. Head

- La testa (head) è la componente finale che effettua le previsioni sulle classi degli oggetti e le loro rispettive bounding box. La testa prende in input le caratteristiche elaborate dal neck e produce output sotto forma di coordinate delle bounding box e probabilità di appartenenza a una classe specifica.
- YOLOv5 utilizza una tecnica chiamata anchor-based prediction, che prevede l'utilizzo di ancore predefinite (bounding boxes di riferimento) per migliorare la precisione nel predire la posizione degli oggetti.

Questa struttura a tre componenti permette a YOLOv5 di essere estremamente efficiente e preciso, rendendolo capace di operare in tempo reale anche su dispositivi con risorse limitate. Il modello è stato progettato per mantenere un equilibrio ottimale tra velocità di elaborazione e accuratezza nel rilevamento degli oggetti, rendendolo uno degli strumenti più utilizzati per applicazioni pratiche di visione artificiale.

3.5.3 Come è stato utilizzato YOLOv5?

You Only Look Once (YOLO) è stato utilizzato per rilevare e classificare geroglifici specifici all'interno di un dataset di immagini annotate.

L'addestramento è iniziato con la configurazione del processo attraverso un file Yet Another Markup Languange (YAML), che ha definito in dettaglio i percorsi dei dati, le classi di oggetti da rilevare e altri parametri essenziali. Questo file ha garantito che l'intero processo fosse chiaro e ripetibile. Un modello YOLOv5s

pre-addestrato è stato scelto come base, sfruttando un modello già allenato su un vasto dataset come ImageNet. Successivamente, è stato applicato il **fine-tuning** per adattare il modello ai dati specifici del progetto. Questo approccio di **transfer learning** ha permesso di ridurre significativamente i tempi di addestramento, sfruttando le conoscenze già acquisite dal modello e migliorandone ulteriormente le prestazioni sui nuovi dati.

Per ottimizzare l'addestramento, sono stati selezionati attentamente diversi parametri, come il numero di epoch, fissato a 100 per garantire un apprendimento adeguato, e la dimensione del batch, mantenuta a 4 immagini per iterazione. Questa scelta ha ottimizzato l'uso della memoria della GPU, garantendo al contempo che il modello potesse essere addestrato in modo efficiente. Anche la risoluzione delle immagini è stata un fattore chiave: è stata impostata a **320x320 pixel**, un compromesso ben ponderato tra il livello di dettaglio delle immagini e la velocità di elaborazione. Questa configurazione ha permesso di mantenere alta la velocità di addestramento, assicurando comunque che il modello fosse in grado di rilevare efficacemente oggetti di varie dimensioni.

Grazie a questo approccio personalizzato, YOLOv5 è stato in grado di adattarsi in modo ottimale ai dati specifici del progetto, migliorando significativamente la sua capacità di rilevare e classificare con precisione gli oggetti di interesse. Questo processo ha portato allo sviluppo di un modello robusto e accurato, capace di operare efficacemente anche in scenari complessi o con variazioni nei dati di input, rendendolo uno strumento estremamente utile per le applicazioni previste.

Capitolo 4

Analisi dei Risultati Ottenuti

In questo capitolo vengono presentati e analizzati i risultati ottenuti a fronte dei vari test e studi compiuti sui diversi modelli precedentemente presentati nel capitolo precedente.

4.1 Metriche di Valutazione

Saranno analizzate due tipologie di metriche: **metriche di performance** e **metriche energetiche** ottenute tramite il tool *CodeCardon*[10].

Le metriche di performance che saranno analizzate sono:

- **F1-Score** - La F1-Score rappresenta la media armonica tra precisione e recall, e può essere particolarmente utile quando il dataset è sbilanciato.
- **Accuracy** - L'accuratezza misura la proporzione di predizioni corrette sul totale, ma può essere fuorviante quando le classi sono fortemente sbilanciate.
- **Recall** - La recall (o sensibilità) indica la capacità del modello di identificare correttamente tutte le istanze positive.
- **Precision** - La precisione misura la proporzione di istanze positive correttamente classificate sul totale delle istanze classificate come positive.

Le metriche energetiche che saranno analizzate sono:

- **Tempo di addestramento** - Rappresenta il tempo totale impiegato dal modello per completare il processo di addestramento.
- **Energia GPU** - Misura il consumo energetico complessivo della GPU durante l'addestramento del modello.
- **Energia CPU** - Indica l'energia consumata dalla CPU nel corso dell'esecuzione del modello.
- **Energia RAM** - Misura l'energia utilizzata dalla RAM per mantenere i dati in memoria durante l'addestramento.
- **Energia totale** - Rappresenta la somma complessiva di tutte le fonti di consumo energetico (GPU, CPU e RAM) durante l'intero processo di addestramento.

La configurazione hardware, su cui sono stati eseguiti gli esperimenti, è composta dai seguenti componenti tecnici:

- **CPU:** Ryzen 5 7600 (TDP = 65W)
- **GPU:** NVIDIA GeForce RTX 4060Ti con 16 GB di RAM dedicata (TDP = 160W)
- **RAM:** 32 GB

4.1.1 CodeCarbon

CodeCarbon è uno strumento open-source progettato per monitorare e quantificare l'impatto ambientale dei progetti di Machine Learning in termini di emissioni di carbonio. Questo strumento è particolarmente utile per comprendere e ridurre l'impronta ecologica associata all'addestramento e all'uso di modelli di apprendimento automatico. Questo è supportato da studi che evidenziano l'importanza di monitorare il consumo energetico per promuovere pratiche sostenibili nell'Artificial Intelligence (Debus et al., 2023[11]).

La sua integrazione nel nostro progetto di guida autonoma è cruciale per diverse ragioni:

- **Consapevolezza Ambientale:** L'uso di **CodeCarbon** ci consente di quantificare l'impatto ambientale del nostro modello, promuovendo una maggiore consapevolezza riguardo all'energia consumata e alle emissioni di carbonio prodotte durante il processo di addestramento.
- **Riduzione dell'Impronta di Carbonio:** I dati forniti ci permettono di adottare strategie per ridurre le emissioni di CO₂. Ad esempio, *Husom et al. (2024)* hanno dimostrato come l'integrazione di CodeCarbon nelle pipeline di Machine Learning possa monitorare e ridurre le emissioni di carbonio, risultando in un modello più ecologico e sostenibile [12].
- **Documentazione e Trasparenza:** I report generati aggiungono un livello di trasparenza alla documentazione del progetto, rendendo esplicativi i costi ambientali e facilitando il rispetto di standard di sostenibilità.

4.2 Metodi di Explainability

Nel contesto dell'apprendimento automatico, la capacità di interpretare e comprendere le decisioni prese da un modello è cruciale, soprattutto in applicazioni critiche. Per questo motivo, è stato adottato Local Interpretable Model-agnostic Explanations (LIME)[13] come metodo di explainability per analizzare e interpretare le predizioni effettuate dai modelli addestrati.

LIME è una tecnica che permette di spiegare le predizioni di qualsiasi modello "black box" generando una spiegazione interpretabile a livello locale, come spiega l'articolo "*Why Should I Trust You?*"[14]", è importante capire come vengono generate le predizioni di un modello di Machine Learning a fine di poterle considerare attendibili. Il principio di base di LIME è quello di approssimare il modello complesso tramite un modello interpretabile (ad esempio, un modello lineare) nella vicinanza della singola predizione che si desidera spiegare. Questo processo avviene in tre fasi principali:

- **Campionamento locale:** LIME genera una serie di nuove istanze variando leggermente l'input originale. Per ognuna di queste varianti, viene registrata la predizione del modello complesso.
- **Ponderazione:** Le nuove istanze vengono pesate in base alla loro somiglianza all'istanza originale, assicurando che il modello interpretabile si concentri sulle regioni rilevanti.
- **Costruzione del modello interpretabile:** Viene addestrato un modello interpretabile (come una regressione lineare o un albero decisionale) sui dati perturbati e ponderati, con lo scopo di approssimare il comportamento del modello complesso in quella specifica regione.

In questo lavoro, LIME è stato utilizzato per fornire spiegazioni visive delle predizioni effettuate dai modelli di classificazione delle immagini. Per ogni immagine testata, sono state generate delle maschere di spiegazione che evidenziano le caratteristiche dell'immagine maggiormente rilevanti per la decisione del modello.

Queste spiegazioni sono state poi confrontate tra vari modelli per analizzare come ciascun modello interpreti gli input e quali siano le differenze principali nelle loro predizioni.

4.2.1 GlyphNet

Analisi dei risultati

Grid Search Nel processo di ottimizzazione del modello *GlyphNet*, sono stati sperimentati vari parametri per ottenere la combinazione ottimale di tasso di apprendimento (*learning rate*) e dimensione del batch (*batch size*). I migliori parametri trovati sono stati un *learning_rate* di 0.001, un *batch_size* di 32, come mostrato dalla griglia di parametri utilizzata per la ricerca (*param_grid*), ciascun addestramento è stato effettuato per un totale di 20 epoche:

```
param\_grid = {
    'learning\_rate': [0.01, 0.001, 0.0001],
    'batch\_size': [32, 64, 128]
}
```

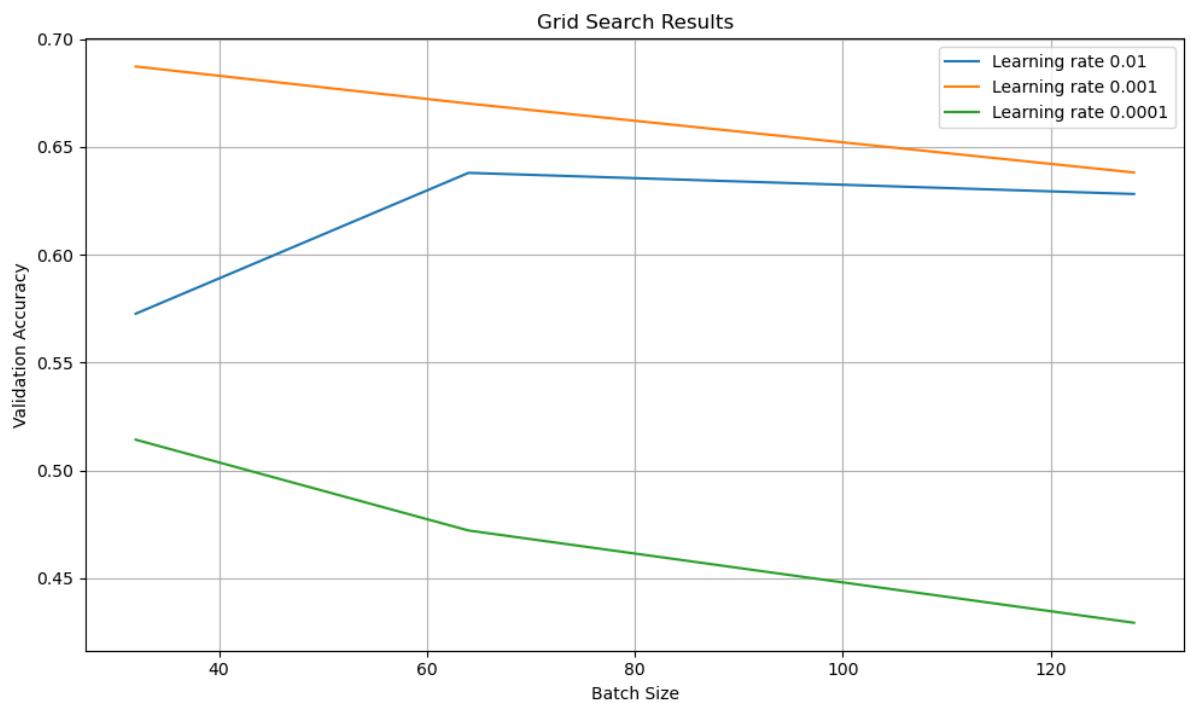


Figura 4.1: Grid search dei parametri per *GlyphNet*.

Validazione della Perdita e Accuratezza Durante l’allenamento, la perdita di validazione è diminuita costantemente, il che indica che il modello stava imparando a fare previsioni più accurate sui dati di validazione. Questo è un segno positivo, poiché una perdita più bassa implica meno errori. L’accuratezza, che rappresenta la percentuale di previsioni corrette, è aumentata progressivamente, partendo da valori iniziali intorno al 28-34% fino a superare il 50% nelle ultime epoche.

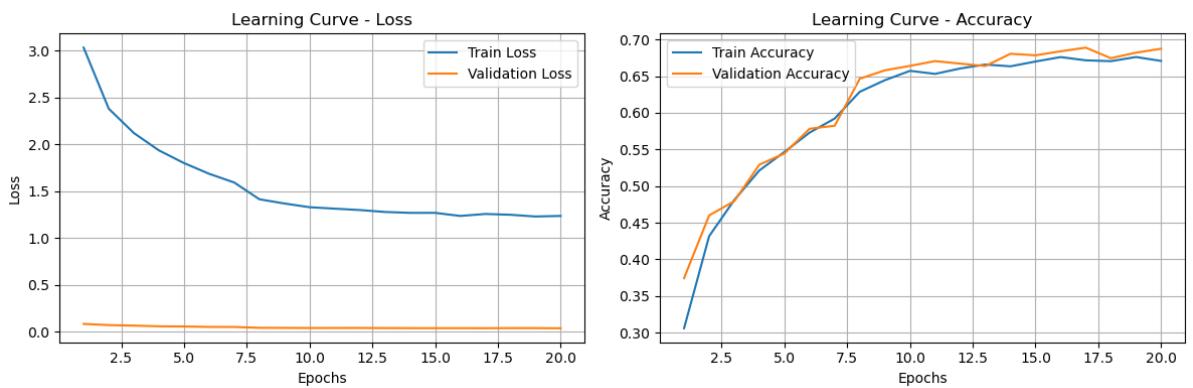


Figura 4.2: Curva di apprendimento del modello *GlyphNet*.

Effetto del Tasso di Apprendimento e della Dimensione del Batch Dai risultati ottenuti, sembra che i valori di tasso di apprendimento più bassi, come 0.001 e 0.0001, abbiano portato a una riduzione più graduale della perdita e a un miglioramento più stabile dell’accuratezza.

La dimensione del batch (*batch size*) influisce sul numero di campioni utilizzati per calcolare l’errore prima dell’aggiornamento dei pesi. Nei nostri esperimenti, dimensioni del batch più piccole, come 32 e 64, hanno portato a miglioramenti più stabili e graduali in termini di accuratezza, mentre dimensioni più grandi hanno introdotto una maggiore variabilità.

Andamento durante le Epoche L'accuratezza all'inizio dell'allenamento era bassa, come previsto. Tuttavia, con il progredire delle epoche, il modello ha continuato a migliorare, raggiungendo un punto di stabilizzazione intorno alla epoca 10, suggerendo che il numero di epoche era eccessivo ciò vuol dire che eventuali miglioramenti e ottimizzazioni del modello avrebbero portato ad un graduale miglioramento prestazionale

Valutazione sul Set di Test I risultati hanno mostrato una *accuracy* di 4.26%, una *precision* di 0.20%, un *recall* di 1.00% e un *F1 Score* di 0.19%.

Metriche Energetiche

Di seguito riportiamo i risultati per le metriche di energetiche dell'intero processo di grid search:

- **Energia GPU** - 0.2366 kWh
- **Energia CPU** - 0.1114 kWh
- **Energia RAM** - 0.0306 kWh
- **Energia totale** - 0.3786 kWh
- **Tempo di addestramento** - 2 ore, 37 minuti e 18.43 secondi.

Metriche di Explainability

L'immagine fornita mostra tre pannelli che confrontano l'immagine originale di un geroglifico con due diverse spiegazioni generate utilizzando tecniche di interpretabilità. Questi pannelli permettono di visualizzare quali aree dell'immagine hanno influenzato maggiormente la decisione del modello *GlyphNet* durante la classificazione.

- **Original Image:** Il primo pannello mostra l'immagine originale del geroglifico che è stata utilizzata come input per il modello *GlyphNet*. Questa immagine serve come riferimento per le successive spiegazioni. Sono stati utilizzati come esempio il geroglifico Aa26 4.3 e D1 4.4.
- **Custom Explanation:** Il secondo pannello rappresenta una spiegazione generata utilizzando un metodo personalizzato. Il metodo custom evidenzia quelle che sono le regioni di interesse e il focus del modello sull'immagine. Per la scarsa qualità del dataset, avendo riscontrato i problemi precedentemente discussi, l'explanation ha dato risultati non soddisfacenti non evidenziando alcuna area di interesse in entrambi gli esempi.
- **Local Interpretable Model-agnostic Explanations (LIME):** Il terzo pannello mostra la spiegazione generata utilizzando la libreria Local Inter-

pretable Model-agnostic Explanations (LIME). Questa explanation si è rilevata molto sensibile ai bordi che sono stati generati tramite la procedura di Data Augmentation. L'esempio 2 mostra che anche in questo caso nessuna area di interesse è stata rilevata, mentre per l'esempio 1 il metodo ha rilevato come "importanti" le aree dove è presente il bordo post-augmentation portando anche in questo caso a risultati non soddisfacenti.

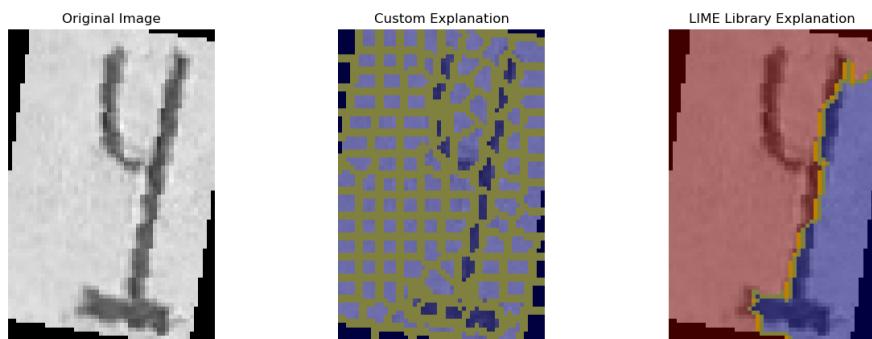


Figura 4.3: Spiegazione modello *GlyphNet*, esempio 1.

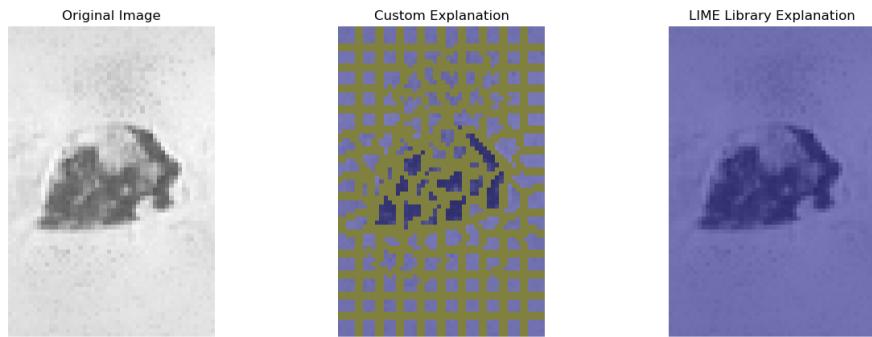


Figura 4.4: Spiegazione modello *GlyphNet*, esempio 2.

4.2.2 ATCnet

Analisi dei risultati

Grid Search Nel processo di ottimizzazione del modello **ATCNet** i parametri migliori identificati sono stati un *learning rate* di 0.001 e una *batch size* di 128. La ricerca dei parametri è stata condotta utilizzando una griglia di parametri (*param_grid*) e ciascun addestramento come nel caso di GlyphNet è stato eseguito per un totale di 20 epoche:

```
param_grid = {
    'learning_rate': [0.01, 0.001, 0.0001],
    'batch_size': [32, 64, 128]
}
```

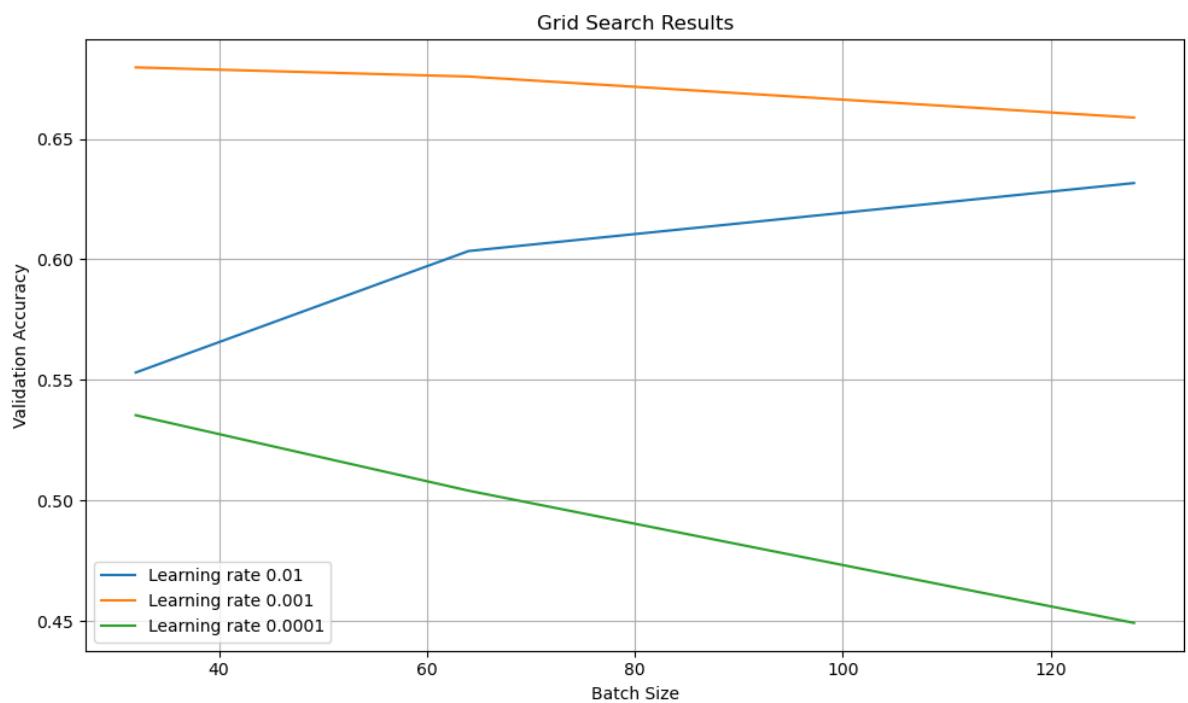


Figura 4.5: Grid search per *ATCNet*.

Validazione della Perdita e Accuratezza Durante l’allenamento, la perdita di validazione è diminuita costantemente, indicando che il modello stava apprendendo a fare previsioni più accurate sui dati di validazione. L’accuratezza, rappresentante la percentuale di previsioni corrette, è aumentata progressivamente, passando da valori iniziali intorno al 30-35% fino a superare il 68% nelle ultime epoche.

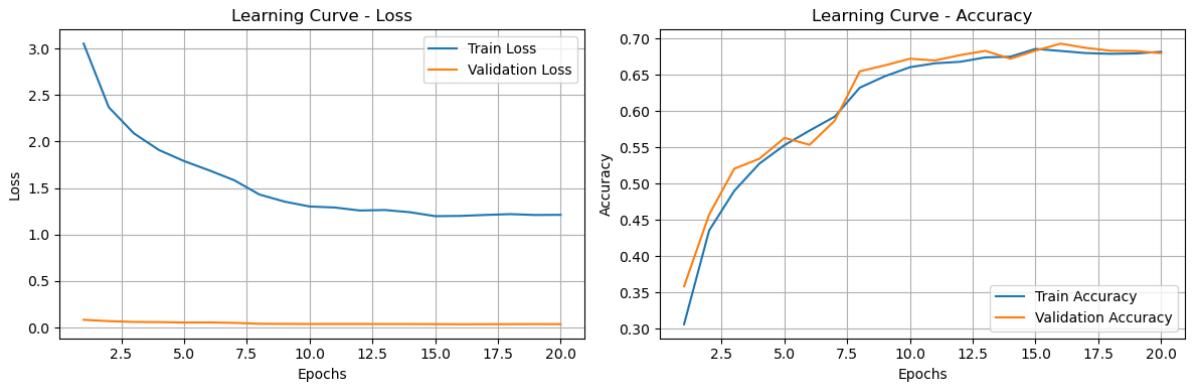


Figura 4.6: Curva di apprendimento del modello ATCNet.

Effetto del Tasso di Apprendimento e della Dimensione del Batch Dai risultati ottenuti, sembra che i valori di tasso di apprendimento, come nel caso di GlyphNet, più bassi, come 0.001 e 0.0001, abbiano portato a una riduzione più graduale della perdita e a un miglioramento più stabile dell’accuratezza.

Nei nostri esperimenti, dimensioni del batch più grandi, come 128, hanno portato a miglioramenti più stabili e graduali in termini di accuratezza, mentre dimensioni più piccole hanno introdotto una maggiore variabilità.

Andamento durante le Epoche All’inizio dell’allenamento, l’accuratezza era bassa, come ci si aspettava. Tuttavia, con il passare delle epoche, il modello ha mostrato un miglioramento continuo, stabilizzandosi verso la fine del processo. Questo indica che il modello potrebbe aver raggiunto un punto di saturazione, suggerendo che eventuali ulteriori miglioramenti potrebbero richiedere ottimizzazioni supplementari.

Valutazione sul Set di Test I risultati hanno mostrato una *precision* del 0.03%, un' *accuracy* di 0.50%, un *recall* di 0.16% e un *F1 Score* di 0.05%.

Metriche Energetiche

Di seguito riportiamo i risultati per le metriche di energetiche dell'intero processo di grid search:

- **Energia GPU** - 0.2388 kWh
- **Energia CPU** - 0.1135 kWh
- **Energia RAM** - 0.0312 kWh
- **Energia totale** - 0.3834 kWh
- **Tempo di addestramento** - 2 ore, 40 minuti e 10.30 secondi.

Metriche di Explainability

L'immagine fornita mostra tre pannelli che confrontano l'immagine originale di un geroglifico con due diverse spiegazioni generate utilizzando tecniche di interpretabilità. Questi pannelli permettono di visualizzare quali aree dell'immagine hanno influenzato maggiormente la decisione del modello *ATCNet* durante la classificazione.

- **Original Image:** Il primo pannello mostra l'immagine originale del geroglifico che è stata utilizzata come input per il modello *ATCNet*. Questa immagine serve come riferimento per le successive spiegazioni. Sono stati utilizzati come esempio il geroglifico Aa26 4.7 e D1 4.8.
- **Custom Explanation:** Il secondo pannello rappresenta una spiegazione generata utilizzando un metodo personalizzato. Anche in questo caso la differenza con la custom explanation di GlyphNet è solo la rappresentazione dei super-pixel, mentre anche questa volta il risultato dell'individuazione delle aree di interesse non è soddisfacente.
- **Local Interpretable Model-agnostic Explanations (LIME):** Il terzo pannello mostra la spiegazione generata utilizzando la libreria Local Interpretable Model-agnostic Explanations (LIME). La spiegazione generata con

LIME ha evidenziato, come nel caso del modello GlyphNet, una sensibilità ai bordi e alle aree di contrasto introdotte durante il processo di Data Augmentation. Nel primo esempio, le aree "importanti" identificate da LIME sono prevalentemente concentrate sui bordi, mentre nel secondo esempio il modello ha rilevato la parte del geroglifico avendo sostanzialmente un risultato di riconoscimento leggermente migliore di GlyphNet.

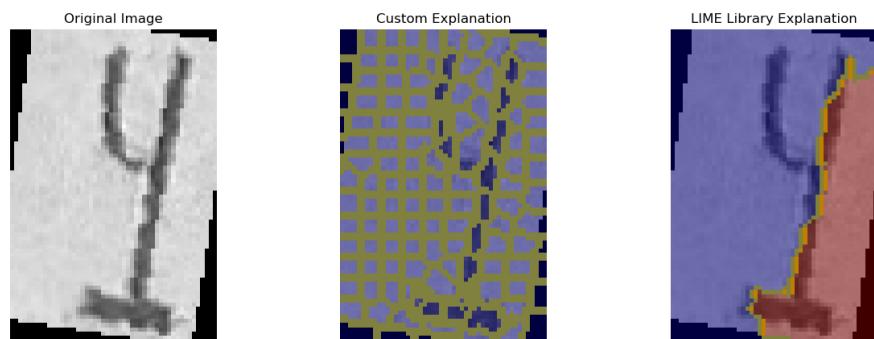


Figura 4.7: Spiegazione modello *ATCNet*, esempio 1.

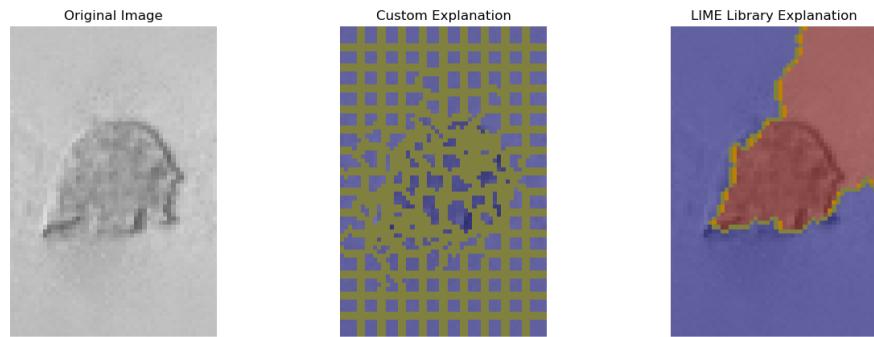


Figura 4.8: Spiegazione modello *ATCNet*, esempio 2.

4.2.3 TResNet

Analisi dei Risultati

Ricerca dei Parametri Ottimali Nel corso dell'ottimizzazione del modello **TResNet**, i parametri ottimali individuati sono stati un *learning rate* pari a 0.0001 e una *batch size* di 16. La selezione dei parametri è stata effettuata tramite una ricerca a griglia (*grid search*) su una serie di configurazioni predefinite (*param_grid*), con ciascun ciclo di addestramento eseguito per un totale di 8 epoche:

```
param_grid = {
    learning_rates: [0.0001, 0.001]
    batch_sizes: [16, 32, 64]
}
```

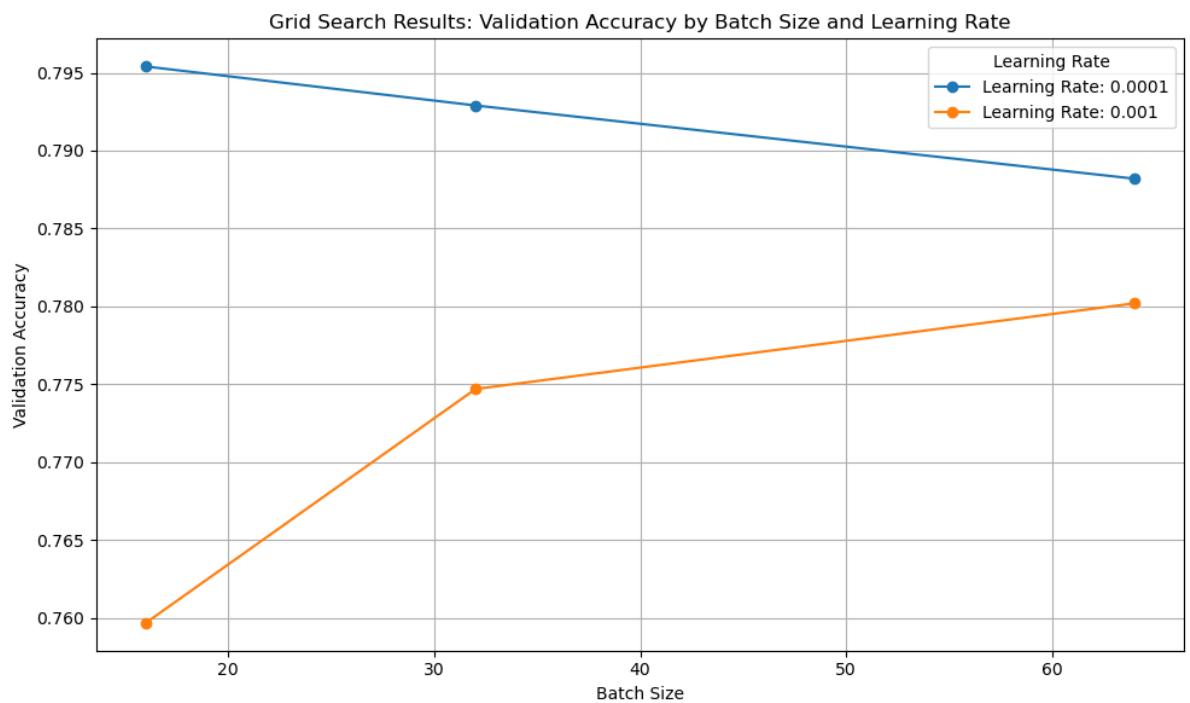


Figura 4.9: Grid search per *TResNet*.

Validazione della Perdita e dell'Accuratezza Durante l'addestramento, si è osservata una costante diminuzione della perdita di validazione, segno che il modello stava migliorando la sua capacità di fare previsioni accurate sui dati di validazione. L'accuratezza del modello è aumentata progressivamente, partendo da un valore iniziale del 43-44% e superando il 79% nelle epoche finali.

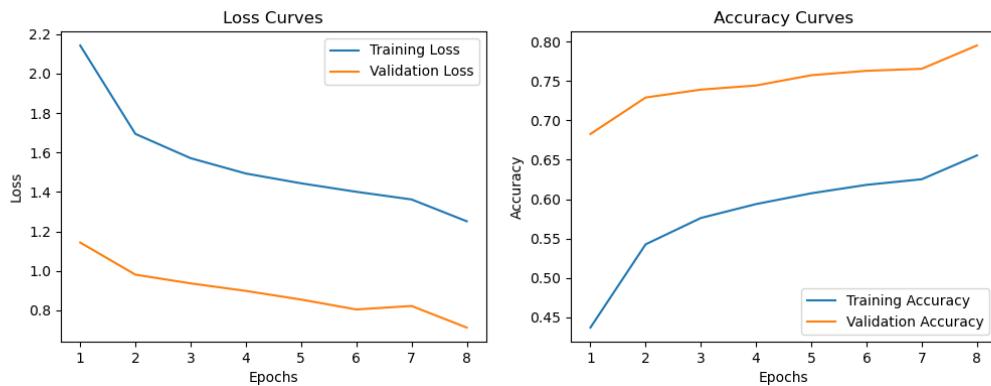


Figura 4.10: Curva di apprendimento del modello *TResNet*.

Influenza del Tasso di Apprendimento e della Dimensione del Batch
Dai risultati ottenuti, emerge che i tassi di apprendimento più bassi, come 0.0001, hanno favorito una riduzione più graduale della perdita e un miglioramento più costante dell'accuratezza.

Nel contesto dei nostri esperimenti, dimensioni del batch più ridotte, come 16, hanno contribuito a miglioramenti più stabili e progressivi dell'accuratezza, mentre dimensioni maggiori hanno introdotto una maggiore variabilità nei risultati.

Progressione durante le Epoche Come atteso, l'accuratezza iniziale dell'allenamento era piuttosto bassa. Tuttavia, con il progredire delle epoche, il modello ha mostrato un miglioramento continuo anche verso la fine.

Valutazione sul Set di Test I risultati finali sul set di test hanno riportato un'accuracy del 79.54%, una precision del 79.61%, un recall del 79.54% e un F1 Score del 78.69%.

Metriche Energetiche

Di seguito riportiamo i risultati per le metriche di energetiche dell'intero processo di grid search:

- **Energia GPU** - 0.8460 kWh
- **Energia CPU** - 0.3128 kWh
- **Energia RAM** - 0.0859 kWh
- **Energia totale** - 1.2448 kWh
- **Tempo di addestramento** - 7 ore, 21 minuti e 37.82 secondi.

Metriche di Explainability

L'immagine fornita mostra tre pannelli che confrontano l'immagine originale di un geroglifico con due diverse spiegazioni generate utilizzando tecniche di interpretabilità. Questi pannelli permettono di visualizzare quali aree dell'immagine hanno influenzato maggiormente la decisione del modello *TResNet* durante la classificazione.

- **Original Image:** Il primo pannello mostra l'immagine originale del geroglifico che è stata utilizzata come input per il modello *TResNet*. Questa immagine serve come riferimento per le successive spiegazioni. Sono stati utilizzati come esempio due diversi geroglifici (con maggior presenza) mostrati nelle figure N35 in figura 4.11 e 4.12.
- **Custom Explanation:** Il secondo pannello rappresenta una spiegazione generata utilizzando un metodo personalizzato. La spiegazione personalizzata mostra una segmentazione dell'immagine in super-pixel, con un livello di dettaglio elevato. Tuttavia, il metodo non risulta pienamente soddisfacente in quanto l'individuazione delle aree di interesse sembra essere, ancora una volta, dispersa su molte piccole regioni, il che potrebbe limitare la chiarezza nell'identificazione delle caratteristiche rilevanti per la classificazione.

- **Local Interpretable Model-agnostic Explanations (LIME):** Il terzo pannello mostra la spiegazione generata utilizzando la libreria Local Interpretable Model-agnostic Explanations (LIME). La spiegazione di LIME evidenzia in modo più chiaro le aree dell'immagine considerate importanti dal modello, concentrandosi in particolare sui bordi e sulle regioni di contrasto, probabilmente introdotte durante il processo di Data Augmentation. Nel primo esempio, LIME evidenzia principalmente i bordi del geroglifico, mentre nel secondo esempio, la spiegazione riesce a isolare meglio la parte significativa del geroglifico, suggerendo un leggero miglioramento nel riconoscimento rispetto al metodo personalizzato.

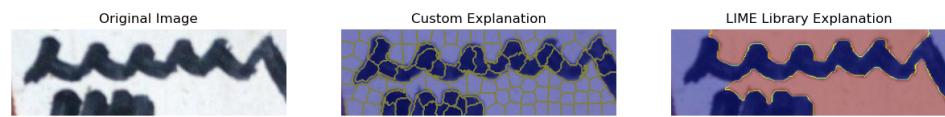


Figura 4.11: Spiegazione modello *TResNet*, esempio 1.

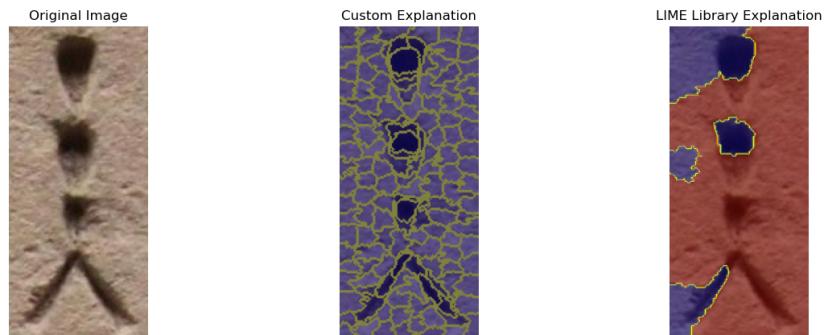


Figura 4.12: Spiegazione modello *TResNet*, esempio 2.

4.2.4 Ensamble Learning

Analisi dei Risultati

Addestramento dei Modelli Il processo di addestramento ha coinvolto la formazione di vari modelli *TResNet*, ciascuno addestrato individualmente su un dataset di geroglifici. I pesi pre-addestrati sono stati caricati dalla libreria Hugging Face, utilizzando una versione sicura dei pesi (*safetensors*) per garantire la sicurezza e l'integrità del processo.

Evoluzione della Perdita e dell'Accuratezza 2 CNN L'addestramento è stato condotto per 10 epoch per ciascun modello, monitorando attentamente la perdita (*loss*) e l'accuratezza su entrambi i set di addestramento e validazione. Nel corso delle epoch, si è osservata una riduzione costante della perdita e un miglioramento progressivo dell'accuratezza, che è passata da circa il 40% all'inizio dell'addestramento fino a superare il 77% nelle ultime epoch. Ad ogni epoch in cui si è registrato un miglioramento nell'accuratezza di validazione, i pesi del modello sono stati salvati.

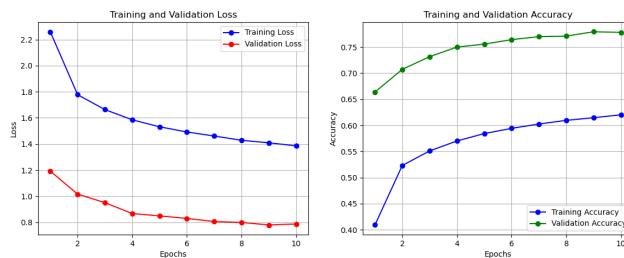


Figura 4.13: Curva di apprendimento per il primo modello *TResNet* a 2 CNN.

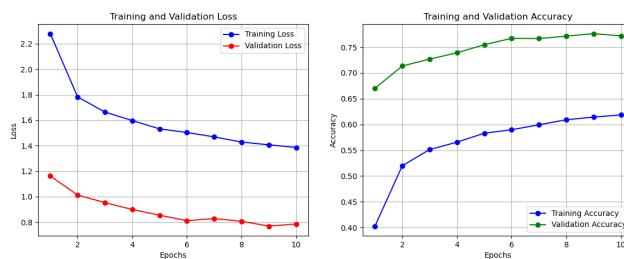


Figura 4.14: Curva di apprendimento per il secondo modello *TResNet* a 2 CNN.

Evoluzione della Perdita e dell'Accuratezza 4 CNN L'addestramento è stato eseguito per 10 epoch per ciascun modello, con un monitoraggio dettagliato della perdita (*loss*) e dell'accuratezza sia sui set di addestramento che di validazione. Nel corso delle epoch, si è osservata una progressiva riduzione della perdita e un miglioramento costante dell'accuratezza, con risultati di validazione che hanno mostrato un aumento fino a circa il 78%. I pesi dei modelli sono stati salvati ogni volta che l'accuratezza di validazione ha mostrato un miglioramento.

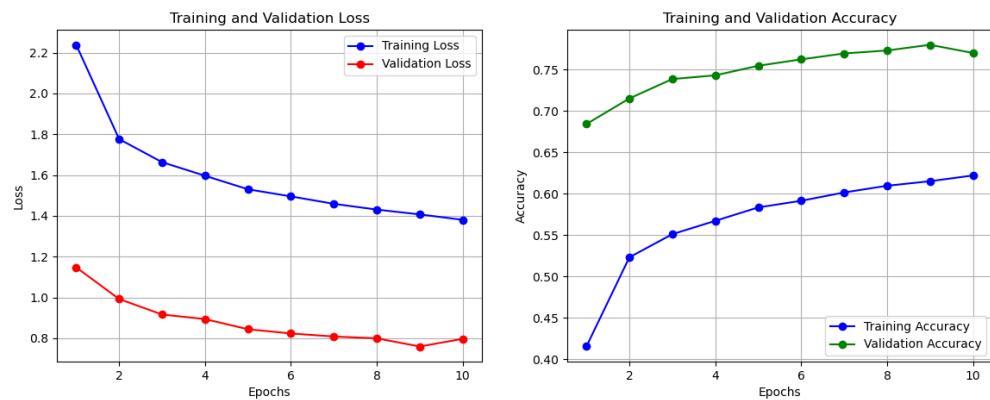


Figura 4.15: Curva di apprendimento per il primo modello *TResNet* a 4 CNN.

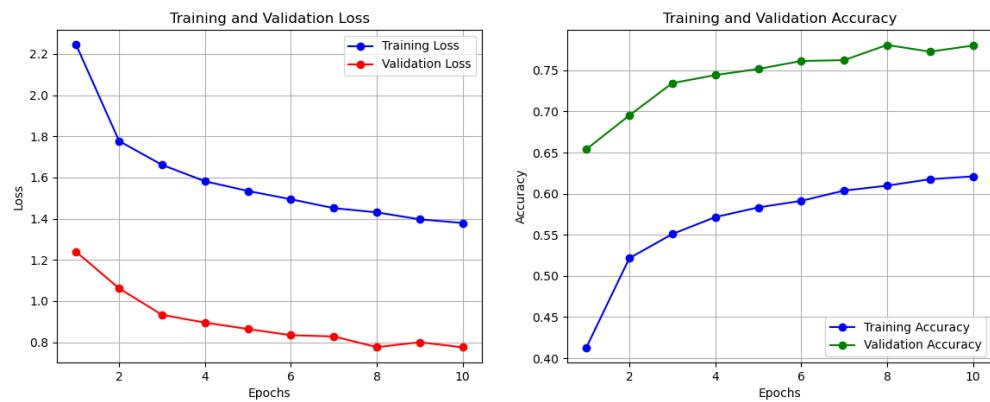


Figura 4.16: Curva di apprendimento per il secondo modello *TResNet* a 4 CNN.

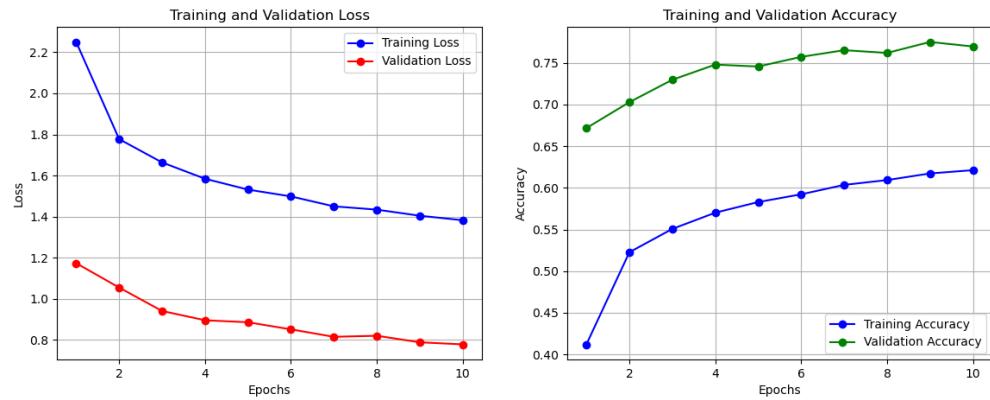


Figura 4.17: Curva di apprendimento per il terzo modello *TResNet* a 4 CNN.

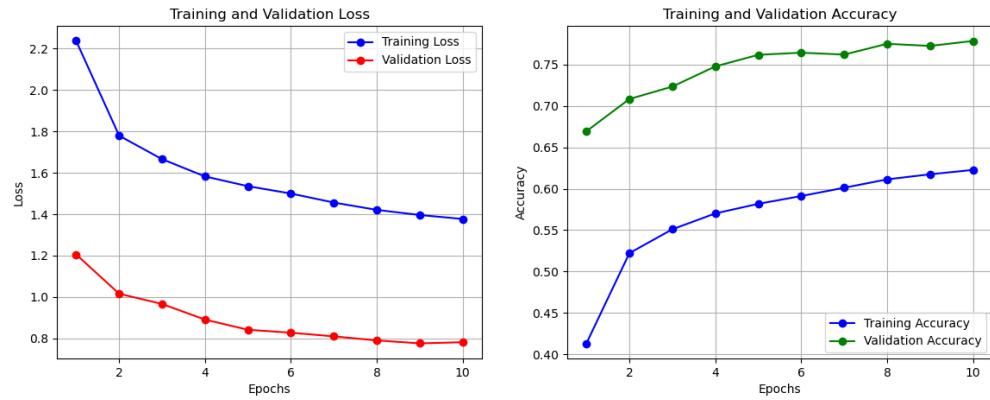


Figura 4.18: Curva di apprendimento per il quarto modello *TResNet* a 4 CNN.

Valutazione delle Prestazioni dell’Ensemble 2 CNN L’ensemble finale ha raggiunto un’accuratezza (*accuracy*) complessiva del 79.29%, con una precisione (*precision*) dell’80.07%, un *recall* del 79.29% e un punteggio F1 (*F1 Score*) del 78.11%. Questi risultati indicano che l’approccio ensemble ha migliorato la capacità di generalizzazione del modello rispetto ai singoli modelli addestrati separatamente.

Valutazione delle Prestazioni dell’Ensemble 4CNN L’ensemble finale ha raggiunto un’accuratezza (*accuracy*) complessiva dell’80.50%, con una precisione (*precision*) dell’81.25%, un *recall* dell’80.50% e un punteggio F1 (*F1 Score*) del 79.53%. Questi risultati evidenziano l’efficacia dell’approccio ensemble, che ha migliorato le capacità di generalizzazione del sistema rispetto a ciascun modello

CNN addestrato individualmente.

Metriche Energetiche

2CNN Di seguito riportiamo i risultati per le metriche di energetiche dell'intero processo di addestramento a 2 CNN:

- **Energia GPU** - 0.3491 kWh
- **Energia CPU** - 0.1275 kWh
- **Energia RAM** - 0.0350 kWh
- **Energia totale** - 0.2015 kWh
- **Tempo di addestramento** - 2 ore, 59 minuti e 59.95 secondi.

4CNN Di seguito riportiamo i risultati per le metriche di energetiche dell'intero processo di addestramento a 4 CNN:

- **Energia GPU** - 0.7016 kWh
- **Energia CPU** - 0.2552 kWh
- **Energia RAM** - 0.0701 kWh
- **Energia totale** - 1.0270 kWh
- **Tempo di addestramento** - 6 ore, 0 minuti e 21.20 secondi.

Metriche di Explainability

Le metriche di explainability esaminate si riferiscono ai migliori risultati ottenuti durante l'addestramento di 4 CNN **TResNet**.

- **Immagine Originale:** Il primo pannello mostra l'immagine originale del geroglifico utilizzata come input per il modello. Questa immagine serve come punto di riferimento per le spiegazioni successive. A titolo di esempio, sono stati scelti due diversi geroglifici (quelli maggiormente presenti), illustrati nelle figure N35 in figura 4.19 e 4.20.
- **Spiegazione Personalizzata:** Il secondo pannello rappresenta una spiegazione ottenuta tramite un metodo personalizzato. Questa spiegazione mostra una segmentazione dell'immagine in super-pixel, con un elevato livello di dettaglio. Tuttavia, il metodo presenta alcune limitazioni, in quanto le aree di interesse individuate risultano distribuite su molte piccole regioni, il che può compromettere la chiarezza nell'identificazione delle caratteristiche rilevanti per la classificazione.
- **Local Interpretable Model-agnostic Explanations (LIME):** Il terzo pannello presenta la spiegazione generata utilizzando la libreria Local Interpretable Model-agnostic Explanations (LIME). In questo caso, LIME evidenzia in modo più chiaro le aree dell'immagine ritenute importanti dal modello. Nel primo esempio, LIME mette in risalto principalmente i bordi del geroglifico, mentre nel secondo esempio, riesce a isolare meglio la parte significativa del geroglifico, suggerendo un miglioramento nel riconoscimento rispetto al metodo personalizzato.

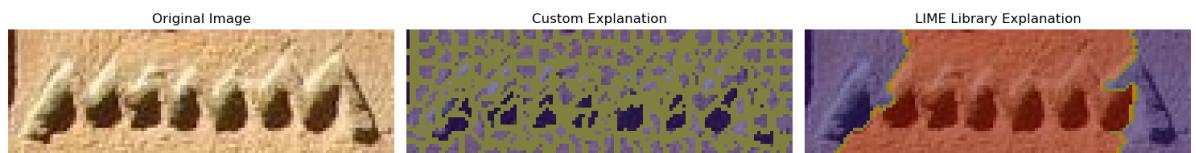


Figura 4.19: Spiegazione modello $\mathcal{A}CNN$, esempio 1.

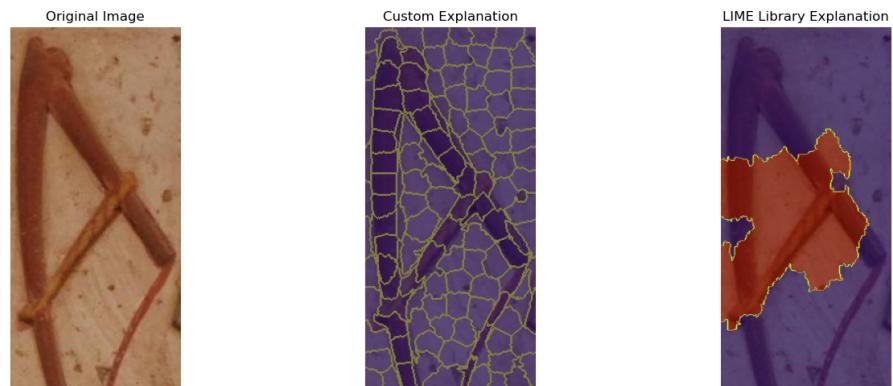


Figura 4.20: Spiegazione modello $\mathcal{A}CNN$, esempio 2.

4.2.5 YOLOv5

Analisi dei risultati

Addestramento del Modello YOLOv5 Durante l’addestramento del modello You Only Look Once (YOLO), sono stati utilizzati diversi parametri e strategie per ottimizzare le prestazioni del modello. Innanzitutto, è stato impiegato un file di configurazione in formato YAML, che ha definito in dettaglio il dataset utilizzato, inclusi i percorsi delle immagini per l’addestramento e la validazione, le classi di oggetti da rilevare e altre impostazioni rilevanti per il processo.

Il numero di epoche per l’addestramento è stato fissato a 100, consentendo al modello di passare più volte sull’intero dataset per migliorare progressivamente le sue capacità di rilevamento e classificazione.

Per ottimizzare l’efficienza dell’addestramento, è stato scelto un batch size di 4. Questo valore rappresenta un compromesso tra l’utilizzo della memoria della GPU e la velocità di addestramento, assicurando che il modello potesse essere addestrato in modo efficace anche con risorse computazionali limitate. Inoltre, la risoluzione delle immagini è stata impostata a 320x320 pixel, bilanciando la necessità di dettagli sufficienti nelle immagini con la necessità di velocità computazionale.

Un elemento fondamentale del processo è stato l’uso di pesi pre-addestrati, specificamente dal modello YOLOv5. Questi pesi, derivati da un modello già addestrato su un dataset più ampio, hanno fornito un punto di partenza solido, permettendo al modello di apprendere più rapidamente e con maggiore efficacia. Questo approccio ha ridotto i tempi di addestramento necessari per raggiungere prestazioni elevate, capitalizzando sull’esperienza pregressa del modello di base.

Perdita e Accuratezza durante l’Addestramento Il processo di addestramento ha coinvolto l’ottimizzazione dei parametri del modello tramite la riduzione delle funzioni di perdita (*loss*). Le curve di apprendimento presentano una chiara riduzione delle perdite, sia per il set di addestramento che per quello di validazione, come mostrato in figura 4.21. Questo indica un miglioramento costante della capacità del modello di adattarsi ai dati e di generalizzare su nuovi dati.

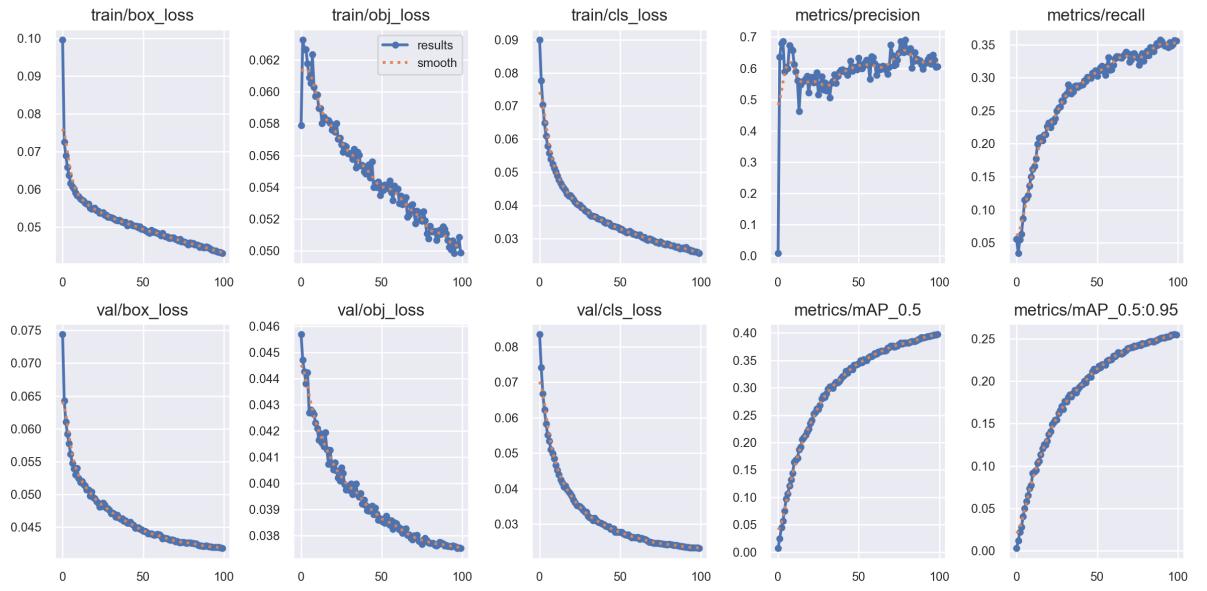


Figura 4.21: Curve di apprendimento e le metriche principali.

Le metriche di precisione e recall mostrano una crescita graduale, suggerendo un miglioramento nella capacità del modello di identificare correttamente gli oggetti di interesse e di minimizzare i falsi positivi. Inoltre, i valori di (mean Average Precision) (mAP) confermano che il modello sta acquisendo una maggiore abilità nella rilevazione accurata degli oggetti, con un incremento continuo durante le epoche di addestramento.

Precision-Confidence Curve La curva Precision-Confidence, rappresentata in figura 4.22, dimostra come la precisione del modello varia al variare della soglia di confidenza. La curva blu indica che la precisione rimane relativamente alta anche per soglie di confidenza elevate, suggerendo che il modello è in grado di mantenere buone prestazioni anche quando è richiesto un alto livello di certezza nelle sue predizioni.

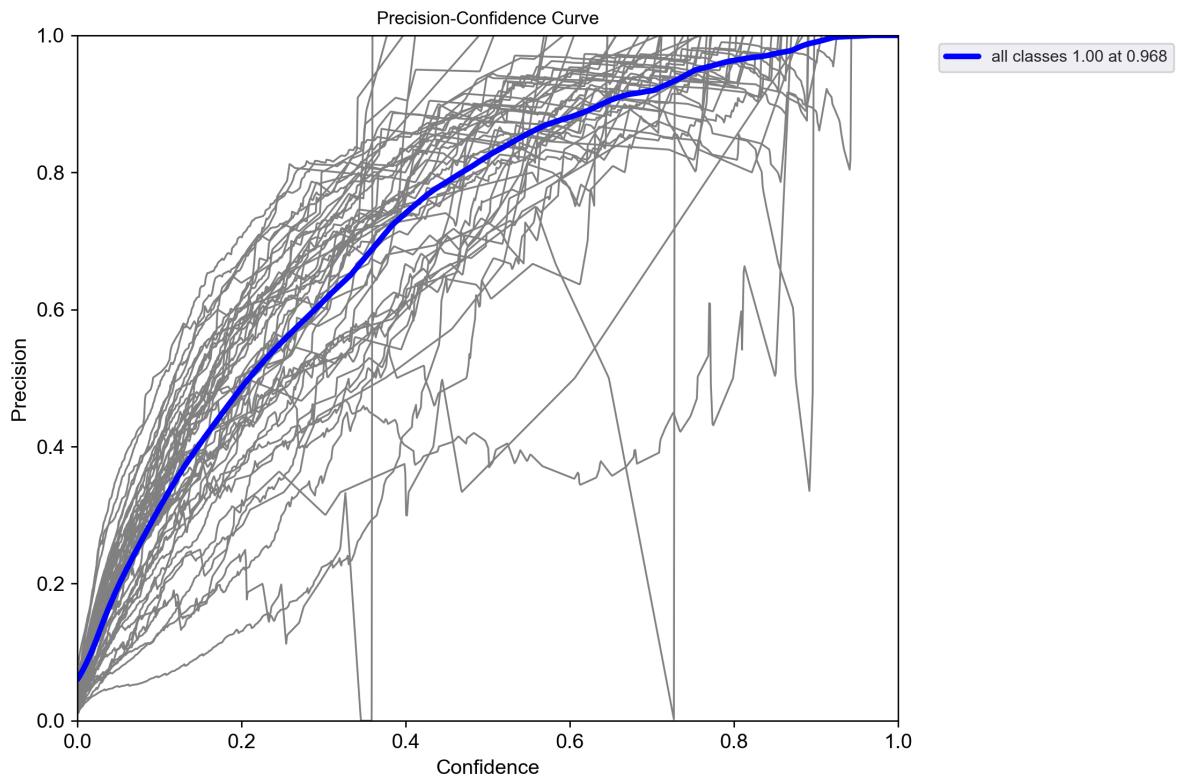


Figura 4.22: Curva di Precisione in funzione della Confidenza.

Precision-Recall Curve La Precision-Recall curve, mostrata in figura 4.23, è cruciale per comprendere il trade-off tra queste due metriche. Un andamento decrescente della precisione all'aumentare del recall è tipico, e nel nostro caso, la curva suggerisce che il modello mantiene una buona capacità di rilevazione (*recall*) mentre la precisione tende a diminuire gradualmente. L'area sotto questa curva, rappresentata dal valore mAP@0.5, evidenzia la necessità di ulteriori ottimizzazioni per migliorare l'equilibrio tra precisione e recall.

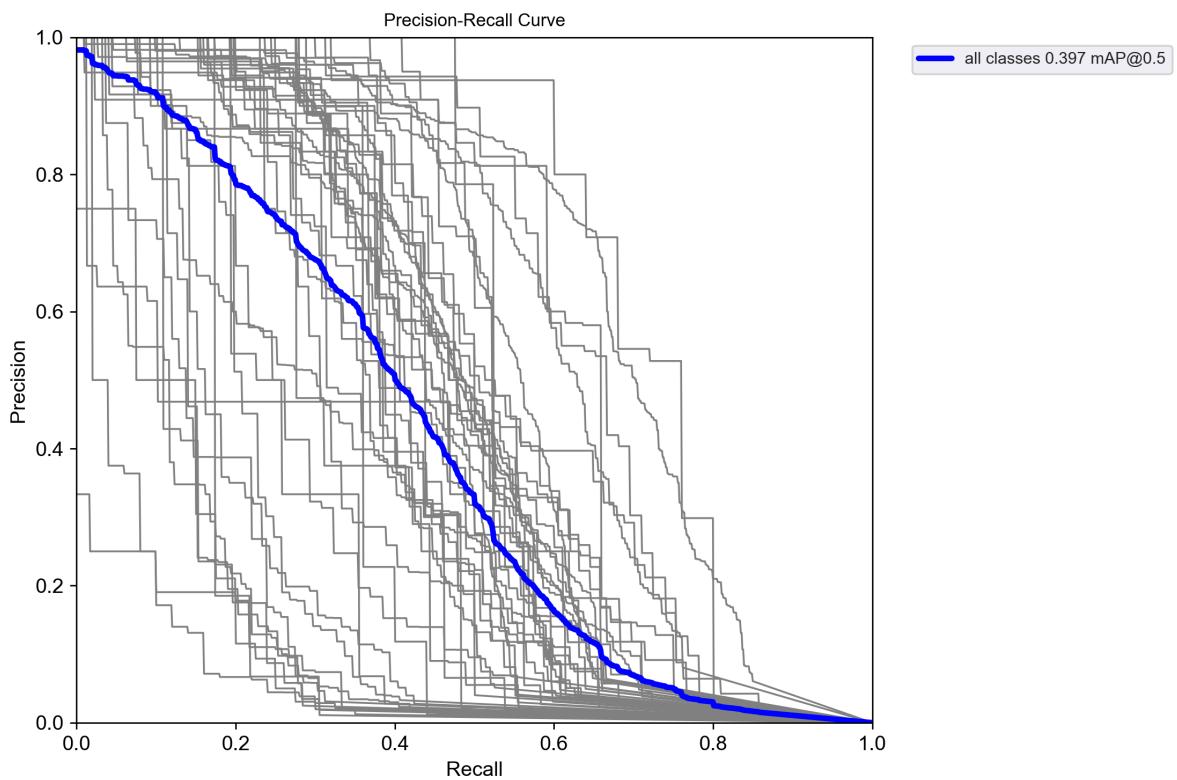


Figura 4.23: Curva Precision-Recall.

Recall-Confidence Curve La curva Recall-Confidence (Figura 4.24) mostra come il richiamo varia con la soglia di confidenza. Come previsto, il richiamo diminuisce con l'aumentare della confidenza, evidenziando il compromesso intrinseco tra questi due parametri. Il modello raggiunge il suo massimo richiamo a soglie di confidenza più basse, ma questo avviene a scapito di una maggiore incertezza nelle predizioni.

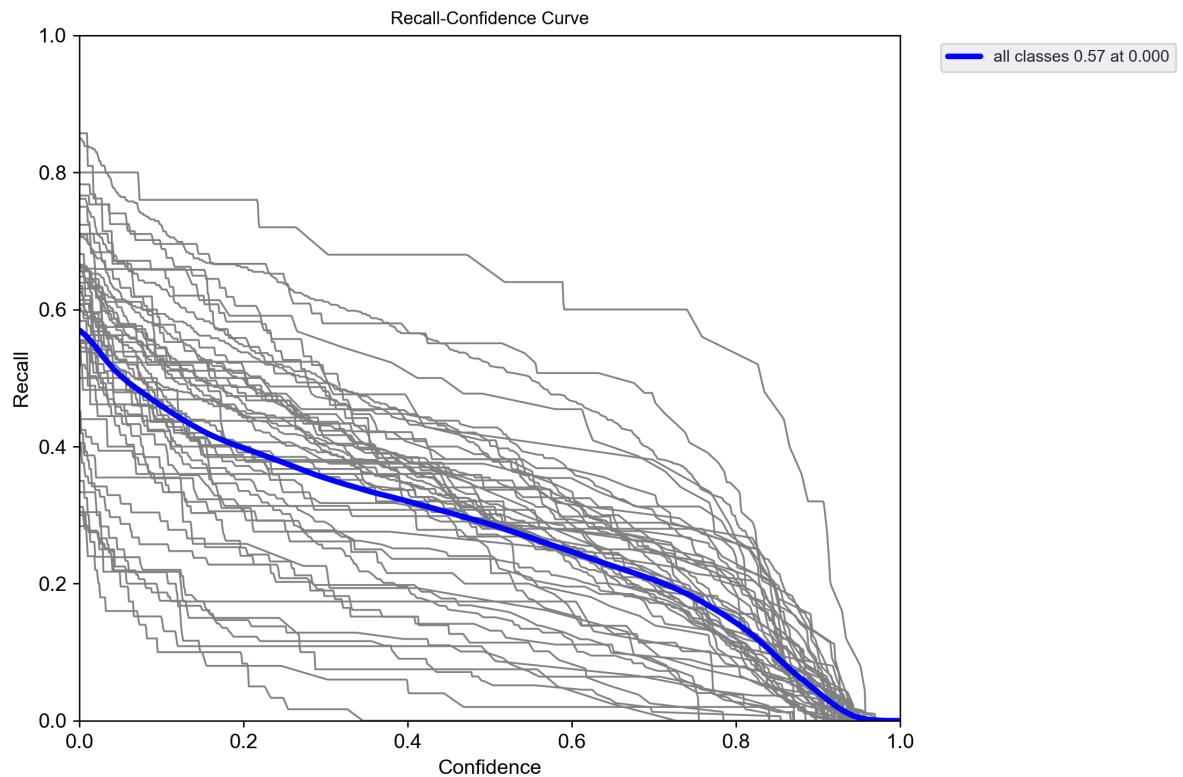


Figura 4.24: Curva di Recall in funzione della Confidenza.

F1-Confidence Curve La curva F1-Confidence (Figura 4.25) rappresenta il punto di equilibrio ottimale tra precisione e richiamo per diverse soglie di confidenza. L'andamento della curva suggerisce che il valore massimo di F1 viene raggiunto a una soglia di confidenza intorno a 0.3, indicando che questa potrebbe essere la soglia ottimale per bilanciare precisione e recall in questo specifico contesto.

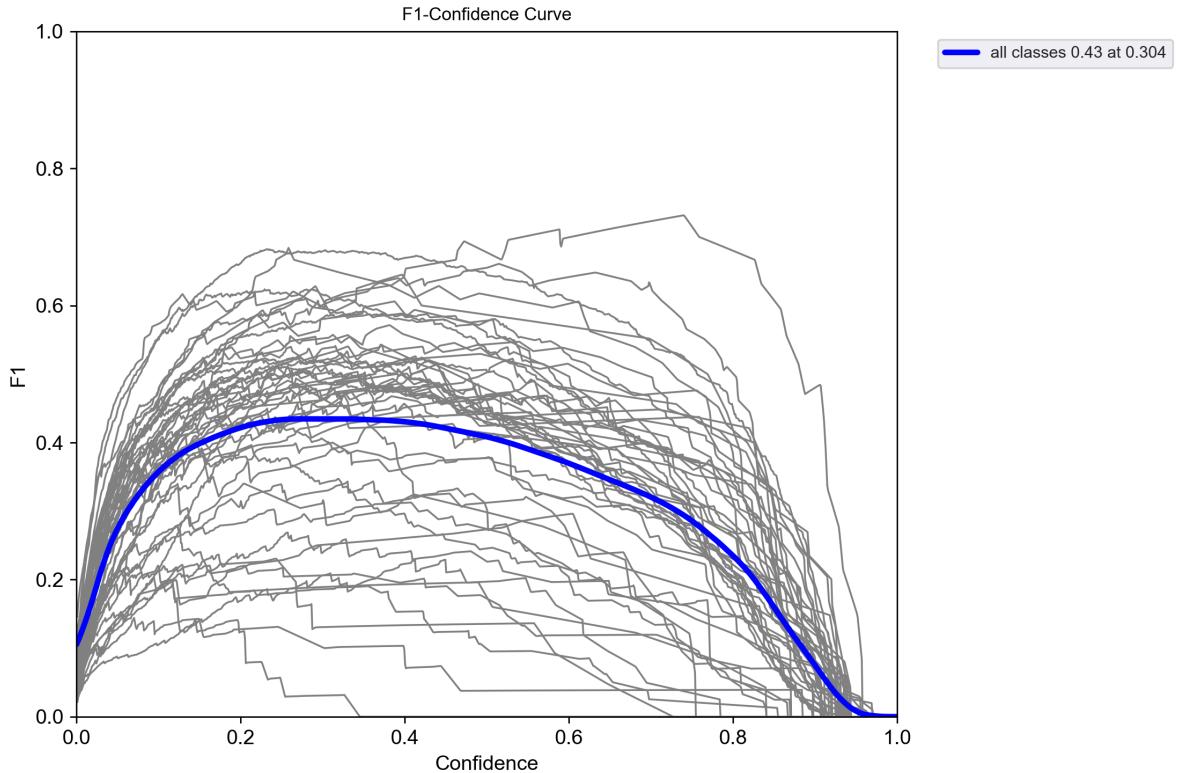


Figura 4.25: Curva F1 in funzione della Confidenza.

Metriche Energetiche

Di seguito riportiamo i risultati per le metriche di energetiche dell'intero processo di addestramento:

- **Energia GPU** - 0.0851 kWh
- **Energia CPU** - 0.0912 kWh
- **Energia RAM** - 0.0251 kWh
- **Energia totale** - 0.2015 kWh
- **Tempo di addestramento** - 2 ore, 8 minuti e 52.99 secondi.

Metriche di Explainability

Il risultato ottenuto tramite il metodo **Grad-CAM**, come illustrato nell'immagine, fornisce una visualizzazione intuitiva delle regioni dell'immagine che hanno influenzato maggiormente le decisioni del modello durante la classificazione dei geroglifici.

- **Immagine Originale (a sinistra)**: Questa rappresenta l'immagine di partenza, che contiene iscrizioni e figure geroglifiche tipiche delle antiche incisioni egizie.
- **Mappa di Attivazione (al centro)**: La seconda immagine mostra la mappa di attivazione prodotta dal modello tramite Grad-CAM. Le aree evidenziate in rosso e giallo indicano le parti dell'immagine che hanno contribuito maggiormente alla predizione finale del modello. In questo caso, le zone con maggiore attivazione sono concentrate intorno ai dettagli più prominenti delle figure e dei simboli geroglifici, suggerendo che il modello si concentra su elementi distintivi come i volti, le mani, e i simboli ben definiti per fare le sue classificazioni.
- **Overlay della Mappa di Attivazione (a destra)**: L'ultima immagine mostra un overlay della mappa di attivazione sull'immagine originale,

combinato con le predizioni del modello, dove si possono vedere i nomi dei simboli riconosciuti e le loro rispettive posizioni. Questo fornisce una chiara indicazione di come il modello identifica e classifica specifici elementi dell'immagine. I riquadri colorati con le etichette (ad esempio "mouth", "hook", "water", ecc.) indicano che il modello ha riconosciuto con successo diversi simboli geroglifici e ha evidenziato le regioni che hanno contribuito a queste predizioni.

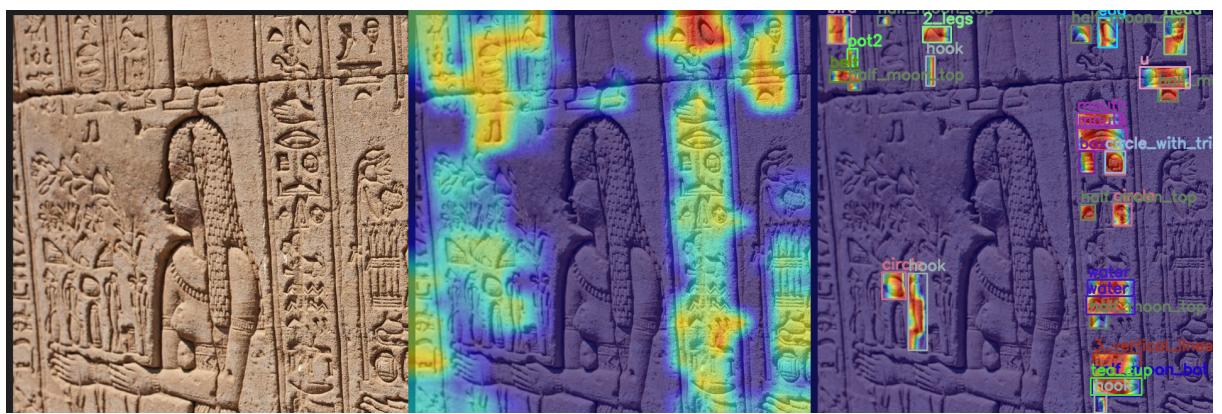


Figura 4.26: Spiegazione modello *YOLO*.

Capitolo 5

Conclusioni

L'idea iniziale del progetto consisteva nel realizzare un blocco unico, che fosse in grado di predire correttamente immagini di geroglifici e generare, al contempo, una didascalia che ne spiegasse il significato. Tramite un approfondito studio della seguente tematica, e delle problematiche riscontrate, si è arrivati a prendere in considerazione diverse soluzioni, tenendo conto della letteratura preesistente (hanno fatto scuola, in questo senso, CNN quali la TResNet). Per quanto concerne, invece, i moduli di explainability, si è scelto di ricorrere sia ad una spiegazione personalizzata, che alla libreria Local Interpretable Model-agnostic Explanations (LIME).

5.1 Risultati a Confronto

In questa sezione verranno mostrati i diversi risultati ottenuti, rilevati tramite le opportune metriche utilizzate.

5.1.1 Metriche di Performance

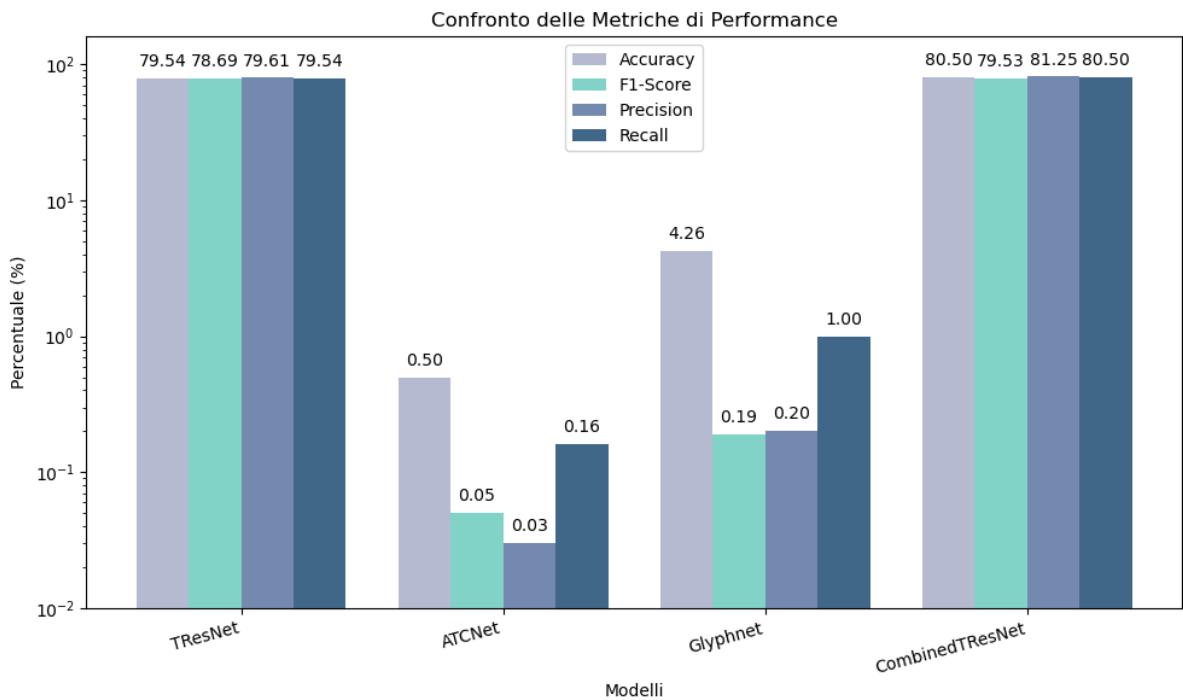


Figura 5.1: Confronto delle metriche di performance.

Il confronto tra le metriche ottenute rende lapalissiana la superiorità della TResNet e della CombinedTResNet (frutto di un Ensamble Learning a 4 CNN) rispetto ad ATCNet e a GlyphNet; in particolare, mentre i risultati dei primi due sono del tutto comparabili (rendendo, di fatto, inutile l'utilizzo preferenziale dell'ensamble, visti i risultati superiori di percentuali quasi nulle), la qualità delle predizioni delle ultime due risulta essere nettamente inferiore, con la GlyphNet che riesce (sebbene di poco) a superare le prestazioni della ATCNet.

5.1.2 Metriche Energetiche

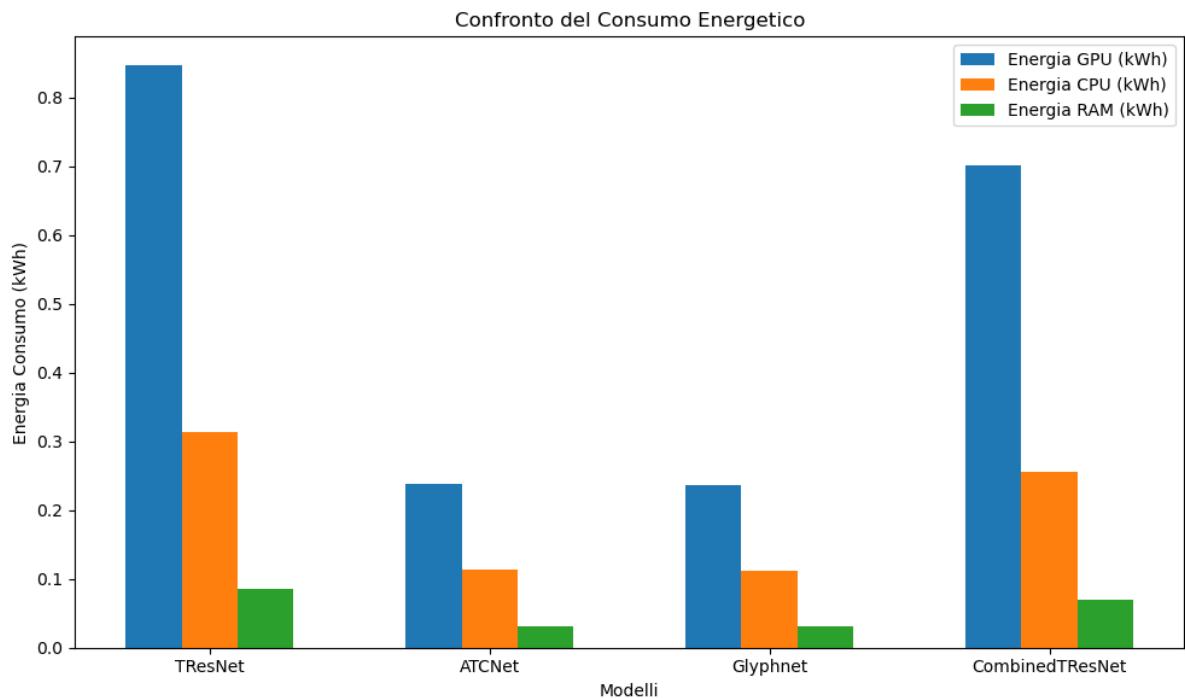


Figura 5.2: Confronto delle metriche energetiche.

Le metriche energetiche mostrano i consumi di GPU, CPU e RAM, e in questo senso, la TResNet si aggiudica lo scettro di rete più energivora (dovuta principalmente alla grid search, che esplora tutto lo spazio delle combinazioni possibili), seguita poi dall'ensamble learning (tenuta ad una distanza per nulla eccessiva). Chiosano, con esiti del tutto comparabili, la ATCNet e la GlyphNet, modelli intrinsecamente più semplici (e per questo, meno esosi da un punto di vista energetico).

5.1.3 Metriche Temporali

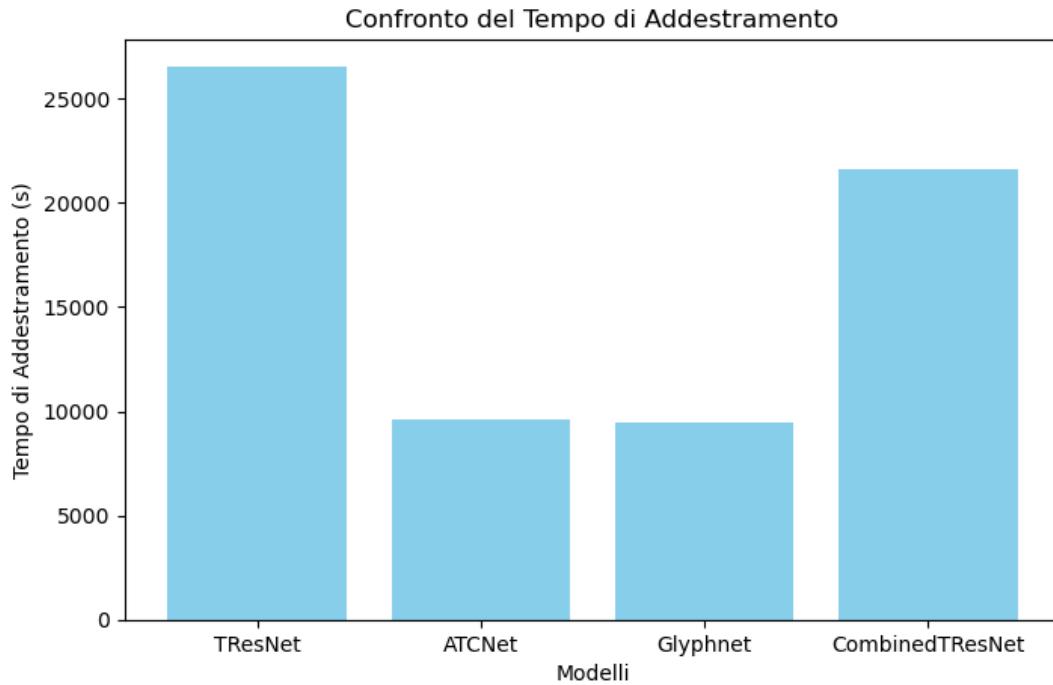


Figura 5.3: Confronto delle metriche temporali.

Per quanto concerne il tempo di addestramento, anche qui, la TResNet risulta essere sul trono (complice, ancora una volta, la grid search); il modello di Ensemble Learning si trova al secondo posto, ed entrambi staccano (con risultati quasi identici) le restanti CNN.

5.2 Risultato migliore

In conclusione, dopo aver fatto numerosi test e studi su varie tecniche di ML abbiamo concluso che il metodo migliore per raggiungere lo scopo prefissatoci è quello di utilizzare una tecnica di Ensemble Learning a 4 CNN seppur leggermente migliore della controparte singola CNN TResNet.

Glossario

Cultural Heritage Il patrimonio culturale comprende i beni culturali, materiali e immateriali, che sono ereditati dal passato e trasmessi alle future generazioni. Include monumenti, siti archeologici, tradizioni e pratiche culturali.

Gardiner's Sign List Il Gardiner's Sign List è un riferimento usato in egittologia per catalogare e classificare i segni geroglifici dell'antico Egitto. Si tratta di un sistema di numerazione sviluppato dal famoso egittologo britannico Sir Alan Gardiner nel suo libro "Egyptian Grammar" pubblicato nel 1927. La lista di Gardiner suddivide i geroglifici in diverse categorie basate su temi come persone, animali, piante, oggetti della vita quotidiana, e simboli più astratti. Ogni segno è classificato con una lettera maiuscola che rappresenta la categoria e un numero che identifica il segno specifico. Ad esempio, "A1" rappresenta l'immagine di un uomo seduto. La Gardiner's Sign List è uno stru-

mento fondamentale per lo studio e la traduzione dei geroglifici egiziani, tuttora ampiamente utilizzato da studiosi e studenti di egittologia.

Otsu Algorithm L'algoritmo di Otsu è una tecnica di sogliatura automatica utilizzata per separare un'immagine in primo piano e sfondo. L'algoritmo cerca di trovare il valore di soglia che minimizza la varianza intraclasse, ovvero la dispersione dei livelli di grigio all'interno delle regioni risultanti.

YOLO (You Only Look Once) YOLO è un modello di deep learning progettato per il rilevamento di oggetti in tempo reale. A differenza delle reti neurali convoluzionali tradizionali, YOLO divide un'immagine in una griglia e prevede bounding box e classi di oggetti per ciascuna cella, permettendo di individuare e classificare più oggetti simultaneamente con alta efficienza e velocità.

Acronimi

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DA	Data Augmentation
DL	Deep Learning
DS	Data Science
EL	Ensamble Learning
FPN	Feature Pyramid Network
LIME	Local Interpretable Model-agnostic Explanations
mAP	(mean Average Precision)
ML	Machine Learning
MLP	Multi-Layer Perseptron
NLP	Natural Language Processing
NN	Neural Network
PANet	Path Aggregation Network
SMOTE	Synthetic Minority Over-sampling Technique
TS	Transfer Learning
YAML	Yet Another Markup Languange
YOLO	You Only Look Once

Riferimenti

- [1] M. Franken, K. Seifi, I. Gkioulekas et al., «Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts,» *ACM*, 2013, This paper introduces a novel approach to automatically recognizing Egyptian hieroglyphs by treating image retrieval as a textual task. DOI: 10.1145/2502081.2502199. indirizzo: <https://dl.acm.org/doi/10.1145/2502081.2502199>.
- [2] E. Hieroglyphs, *Gardiner's Sign List*, Gardiner's sign list is a list of common Egyptian hieroglyphs compiled by Sir Alan Gardiner. It is considered a standard reference in the study of ancient Egyptian hieroglyphs., 1957. indirizzo: <https://www.egyptianhieroglyphs.net/gardiners-sign-list/>.
- [3] A. Barucci, C. Cucci, M. Franci, M. Loschiavo e F. Argenti, «A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification,» *IEEE Access*, 2021, An insightful exploration into how modern deep learning can decode the mysteries of ancient Egyptian hieroglyphs, pushing the boundaries of both technology and history. DOI: 10.1109/ACCESS.2021.3110082. indirizzo: <https://ieeexplore.ieee.org/document/9506887>.
- [4] T. Guidi, L. Python, M. Forasassi et al., «Egyptian Hieroglyphs Segmentation with Convolutional Neural Networks,» *Algorithms*, 2023, An innovative exploration of using convolutional neural networks to segment ancient Egyptian hieroglyphs, blending deep learning techniques with cultural heritage studies. DOI: 10.3390/a16020079. indirizzo: <https://www.mdpi.com/journal/algorithms>.

- [5] A. Ev, *GlyphNet-PyTorch: A PyTorch Implementation of GlyphNet for Hieroglyph Classification*, A PyTorch-based implementation for hieroglyph classification using deep learning., 2023. indirizzo: <https://github.com/alexeyev/glyphnet-pytorch>.
- [6] L. Liu, Q. Yan, J. Li et al., *ACTNet: A Dual-Attention Adapter with a CNN-Transformer Network for the Semantic Segmentation of Remote Sensing Imagery*, An article discussing the implementation and evaluation of the ACTNet model for remote sensing imagery segmentation., 2023. indirizzo: <https://www.mdpi.com/2072-4292/15/9/2363>.
- [7] A. M. Team, *TResNet: High Performance GPU-Dedicated Architecture*, A highly efficient neural network architecture optimized for GPUs., 2020. indirizzo: <https://github.com/Alibaba-MIIL/TResNet>.
- [8] G. Jocher, *YOLOv5: You Only Look Once version 5*, An advanced real-time object detection algorithm., 2020. indirizzo: <https://github.com/ultralytics/yolov5>.
- [9] Adam Paszke and Sam Gross and Soumith Chintala and Gregory Chanan, *PyTorch*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox e R. Garnett, cur., 2019. indirizzo: <https://pytorch.org/>.
- [10] C. team, *CodeCarbon: A tool to estimate and track the carbon emissions produced by AI computing*, A library to estimate the carbon footprint of AI models during training and deployment., 2021. indirizzo: <https://github.com/mlco2/codcarbon>.
- [11] C. Debus, M. Piraud, A. Streit e M. Götz, «Reporting electricity consumption is essential for sustainable AI,» *Nature Machine Intelligence*, 2023, This paper emphasizes the importance of reporting electricity consumption in AI development to ensure sustainability in the field. DOI: 10.1038/s42256-023-00750-1. indirizzo: <https://www.nature.com/articles/s42256-023-00750-1>.

- [12] E. J. Husom, S. Sen e A. Goknil, «Engineering Carbon Emission-aware Machine Learning Pipelines,» *ACM Transactions on AI Engineering*, 2024, A comprehensive study on developing machine learning pipelines that consider carbon emissions, advancing environmentally responsible AI engineering. DOI: 10.1145/3644815.3644943. indirizzo: <https://doi.org/10.1145/3644815.3644943>.
- [13] C. G. Marco Tulio Ribeiro Sameer Singh, *LIME: Local Interpretable Model-Agnostic Explanations*, A framework for explaining the predictions of machine learning models., 2016. indirizzo: <https://github.com/marcotcr/lime>.
- [14] M. T. Ribeiro, S. Singh e C. Guestrin, «“Why Should I Trust You?”: Explaining the Predictions of Any Classifier,» *ACM Transactions on Knowledge Discovery from Data*, 2016, A foundational paper introducing LIME, a method for explaining predictions made by any classifier. DOI: 10.1145/2939672.2939778. indirizzo: <https://doi.org/10.1145/2939672.2939778>.