Riconoscimento di Azioni con Armi da Fuoco nei Video

Giulio Vignati[1,2*] and Matteo Losa[2,3*]

*Corresponding author(s). E-mail(s): s1120527@studenti.univpm.it; s1119369@studenti.univpm.it;

Abstract

In questo studio, abbiamo testato diversi approcci di video classification utilizzando l'architettura MoViNet (Mobile Video Networks) [1], una famiglia di modelli di video classification ottimizzati per dispositivi mobili, concentrandoci sul riconoscimento delle azioni connesse all'uso di armi da fuoco. In particolare, è stata testata la versione MoViNet-A0, ovvero la versione più leggera che meglio si adatta a dispositivi con risorse limitate. Sono state utilizzate varie etichettature dei dati per ottimizzare le performance del modello. Inoltre, è stata valutata un'architettura basata su trasformatori per lo stesso compito, portando risultati altrettanto validi.

1 Introduzione

L'identificazione tempestiva di oggetti pericolosi, come le armi da fuoco, all'interno dei video è di fondamentale importanza per mitigare potenziali danni. La capacità di rilevare rapidamente e accuratamente tali oggetti è cruciale in numerosi contesti di sicurezza e sorveglianza.

Questo progetto si basa su un dataset di video [2], ciascuno dei quali rappresenta un individuo impegnato in azioni specifiche di videosorveglianza. Il compito principale è la classificazione video per il riconoscimento delle azioni legate all'uso di armi da fuoco. Questo tipo di riconoscimento è essenziale per migliorare le applicazioni di sicurezza e sorveglianza, consentendo una risposta più efficace in situazioni di potenziale pericolo.

La sfida di riconoscere le azioni legate all'uso di armi da fuoco nei video richiede tecniche di deep learning avanzate per analizzare efficacemente i video in input e identificare i pattern associati a tali azioni. In questo contesto, l'architettura MoViNet rappresenta una soluzione all'avanguardia, ottimizzata per l'efficienza computazionale e l'accuratezza nel riconoscimento delle azioni.

In particolare, per la classificazione video, abbiamo utilizzato la versione MoViNet-A0-Base, che implementa convoluzioni 3D standard senza l'utilizzo di stream buffer, pre-addestrata sul vasto dataset di action recognition Kinetics-600. MoViNet-A0 è progettata per essere altamente efficiente, rendendola adatta all'implementazione su dispositivi con risorse computazionali limitate, come quelli utilizzati in molte applicazioni di sorveglianza in tempo reale.

Oltre a MoViNet-A0, è stata valutata anche un'architettura basata su trasformatori per il medesimo compito. Questa utilizza un diverso approccio per ricostruire il significato dei frame temporali. I risultati sono stati molto validi in entrambi i casi.

2 Analisi dello stato dell'arte

Il riconoscimento di azioni legate all'uso di armi da fuoco nei video rappresenta una sfida significativa nel campo della visione artificiale e del deep learning. Questo problema è di particolare rilevanza per le applicazioni di sicurezza e sorveglianza, dove la capacità di identificare comportamenti pericolosi è di cruciale importanza. In questo capitolo, verranno esaminati e valutati i metodi e le tecnologie attuali utilizzati per affrontare questo problema oltre a qualche richiamo degli approcci più all'avanguardia per la detection di armi.

Diverse architetture di deep learning sono state sviluppate per il riconoscimento delle azioni nei video. Di seguito, vediamo le più rilevanti.

2.1 Reti Convoluzionali 3D

Le reti convoluzionali 3D (3D-CNN) sono una estensione delle reti convoluzionali 2D (2D-CNN) progettate per lavorare con dati volumetrici, come video, che hanno una dimensione temporale oltre alle dimensioni spaziali. Mentre le 2D-CNN sono efficaci per l'elaborazione di immagini statiche, le 3D-CNN sono particolarmente adatte per catturare informazioni sia spaziali che temporali in sequenze di immagini, come nel caso dei video. A differenza delle convoluzioni 2D, le convoluzioni 3D utilizzano kernel che si muovono lungo tre dimensioni; lo stesso vale per le operazioni di pooling. Tra le 3D CNN più utilizzate abbiamo 3D ResNet, 3D DenseNet e 3D Inception. Nonostante la loro efficacia nel catturare informazioni spazio-temporali dai dati di input, le 3D-CNN possono essere molto computazionalmente intensive e richiedono grandi quantità di dati per l'addestramento, rendendole generalmente inadatte per i dispositivi mobili ed esecuzioni real-time.

2.2 Reti Neurali Ricorrenti

Una rete neurale ricorrente (RNN - recurrent neural network) è un tipo di rete neurale artificiale che utilizza dati sequenziali o dati di serie temporali. A differenza delle reti neurali tradizionali, le RNN hanno connessioni che formano cicli all'interno della rete, permettendo di mantenere una sorta di memoria degli stati precedenti. Lo strato di input riceve i dati in modo sequenziale mentre i neuroni nello strato ricorrente ricevono l'input attuale e l'output dallo stato precedente permettendo alla rete di mantenere informazioni sugli stati passati. Le RNN sono tipicamente addestrate tramite

retropropagazione con cui possono andare in contro al problema del "vanishing" gradient. Il problema del "vanishing gradient" (gradiente che scompare) si verifica nelle reti neurali profonde durante il processo di addestramento con la retropropagazione. Quando i gradienti delle funzioni di perdita sono calcolati e propagati all'indietro attraverso la rete, possono diventare estremamente piccoli in strati vicini all'input. Questo porta a un aggiornamento dei pesi molto lento o nullo, impedendo al modello di apprendere efficacemente. Il problema è particolarmente prevalente nelle reti con molti strati, poiché i gradienti diminuiscono esponenzialmente con la profondità della rete, rendendo difficile l'addestramento di strati profondi. Un tipo di RNN che risolve questo inconveniente è la long-short term memory (LSTM) network. Questo tipo di rete utilizza celle di memoria e meccanismi di porte per controllare il flusso di informazioni, permettendo alla rete di memorizzare e recuperare informazioni su lunghi periodi. Le RNN si rivelano particolarmente efficaci nel gestire dati sequenziali e le loro dipendenze temporali, a discapito di tempi di addestramento piuttosto lunghi.

2.3 Detection di armi

Nonostante il nostro lavoro sia incentrato sulla classificazione video, riportiamo comunque alcuni degli approcci più recenti adottati per i task di detection di armi. In [3] sono state studiate le prestazioni del modello di detection one-stage M2Det [4] addestrato su un dataset di armi catturate da telecamere di sorveglianza CCTV. Il modello è stato poi validato sul dataset di crime video UCF registrando un aumentato la precisione media del rilevamento delle armi del 18% rispetto ai metodi esistenti. Mentre in [5] si esplora l'applicazione del modello di detection YOLO per creare un sistema di rilevamento delle armi da fuoco. In questo paper, gli autori hanno costruito un dataset basato sull'Internet Movie Firearm Database (IMFDB). Il modello utilizzato si è dimostrato efficace nel rilevamento e identificazione di oggetti nelle immagini, producendo talvolta risultati più accurati e coerenti rispetto agli esseri umani. Infine, in [6], è stato testato un modello basato sull'architettura di detection Faster R-CNN per l'attivazione di allarmi di sicurezza. Il modello studiato ha mostrato ottimi risutati anche in video di bassa qualità, attivando l'allarme con successo in 27 su 30 scene in meno di 0,2 secondi.

3 Materiali e metodi

In questa sezione veranno illustrati il materiale di partenza, i metodi e le tecnologie utilizzate per la realizzazione dei test.

3.1 Dataset

Il dataset proposto è composto da 398 video, ciascuno dei quali rappresenta un soggetto impegnato in specifiche azioni di sorveglianza video. Le annotazioni di ground truth per questo dataset sono state presentate in formato JSON standard COCO, offrendo informazioni sul dataset, sui frame dei video e sulle annotazioni, incluse le bounding box precise che delineano gli oggetti rilevati. Tuttavia, per questo studio, non sono

state utilizzate le bounding box poiché ci siamo limitati alla classificazione delle azioni nei video. La suddivisione originale delle classi è la seguente:

- No_Gun: video in cui non sono presenti armi da fuoco.
- Handgun: video in cui il soggetto utilizza una pistola.
- Machine_Gun: video in cui il soggetto maneggia un fucile.

Nel dataset originale, i video di queste classi sono suddivisi a loro volta in sotto-categorie in base all'azione eseguita nel video, alla mano usata, all'obiettivo da mirare, alla posizione del braccio e alla posa della persona. È stato inoltre distino il livello di luminosità, il dispositivo e la posizione di ripresa.

Grazie a queste informazioni aggiuntive, abbiamo pensato di effettuare due diversi tipi di test: il primo è incentrato sulla diversa pericolosità dell'azione presente nel video, andando quindi a distinguere:

- nessuna arma (livello di pericolo basso), label 0.
- arma presente ma non puntata (livello di pericolo medio), label 1.
- arma puntata (livello di pericolo alto), label 2.

Il secondo test è invece incentrato sul tipo di arma rilevata, andando a classificare il video su:

- No_Gun: video in cui non sono presenti armi da fuoco, label 0.
- Handgun: video in cui il soggetto maneggia una pistola, label 1.
- Machine_Gun: video in cui il soggetto maneggia un fucile, label 2.

Nel nostro caso, per lavorare in maniera efficiente sul dataset, abbiamo lavorato con *DataFrame* e *train_test_split* della libreiria *scikit-learn*, in maniera tale da garantire un dataset ben bilanciato. Le etichette sono state caricate in dei file *.csv* contenenti *filename,tag*. Rispetto al dataset originale, abbiamo quindi etichettato l'intero video invece di andare a descrivere ogni frame.

Il dataset è stato suddiviso per il 70% nel training set e il restante è stato diviso per il 50% tra validation set e test set. Nella tabella 1 è rappresentata la distribuzione delle classi per il caso di classificazione del rischio, mentre nella 2 è rappresentato il caso di classificazione dell'arma utilizzata.

	Classe 0	Classe 1	Classe 2	Totale
Train set	87	92	99	278
Validation set	15	18	27	60
Test set	16	22	22	60
Totale	118	132	148	398

Table 1 Distribuzione del dataset per il caso di classificazione del rischio.

	Classe 0	Classe 1	Classe 2	Totale
Train set	84	101	93	278
Validation set	19	23	18	60
Test set	15	17	28	60
Totale	118	141	139	398

Table 2 Distribuzione del dataset per il caso di classificazione dell'arma utilizzata.

3.2 MoViNet

Come già detto, i modelli MoViNet sono una famiglia di reti pensate per la classificazione video efficiente anche in modalità online. MoViNet introduce tecniche innovative per ridurre la complessità computazionale e la memoria richiesta dalle reti neurali convoluzionali 3D (3D-CNN), mantenendo elevate le prestazioni di riconoscimento. Di seguito sono riportate le principali innovazioni che contribuiscono alla sua efficienza:

1. MoViNet search space: MoViNet definisce uno spazio di ricerca per la rete video che permette l'uso della Neural Architecture Search (NAS) per generare architetture 3D-CNN efficienti e diversificate. Questa tecnica consente di ottimizzare le rappresentazioni delle caratteristiche spaziotemporali, bilanciando in modo ottimale l'efficienza computazionale e l'accuratezza del modello.
2. Stream buffer: lo stream buffer è progettato per gestire video in streaming, permettendo l'elaborazione di video in piccoli subclip consecutivi. Il video di input viene suddiviso in subclip più piccoli e gestibili. Ogni subclip viene processato indipendentemente, ma le informazioni temporali vengono mantenute attraverso i subclip mediante lo stream buffer.
3. Temporal ensemble: una tecnica che combina le predizioni effettuate dal modello in diverse fasi dell'addestramento contribuendo così a migliorare le prestazioni generali e a ridurre il rischio di overfitting. Durante l'inferenza, invece, vengono combinate le predizioni del modello per ogni subclip del video in input.

Questi approcci hanno permesso a MoViNet di ottenere ottime prestazioni dal punto di vista di risorse computazionali a discapito di una lieve perdita nell'accuratezza. Ad esempio, MoViNet-A0, la versione più leggera, raggiunge livelli simili di accuratezza rispetto a MobileNetV3 ma con il 75% di FLOPs in meno.

3.3 Transformers per la Video Classification

Gli approcci di video classification basati su transformer sfruttano l'architettura del transformer per elaborare sequenze di frame video. La struttura tipica di un modello transformer per la classificazione video include i seguenti componenti principali:

1. Embedding del frame: questo passaggio spesso comporta l'utilizzo di una rete neurale convoluzionale (CNN) pre-addestrata per estrarre le caratteristiche spaziali dai frame, seguita da una proiezione lineare per adattare le dimensioni delle caratteristiche all'input del transformer.

2. Positional Encoding: gli encoding posizionali vengono aggiunti ai vettori di embedding. Questi encoding forniscono informazioni sulla posizione temporale di ciascun frame nella sequenza video.
3. Transformer: i vettori di embedding risultati dal passo precedente, vengono forniti al layer di transformer. Questi layer consistono in meccanismi di Multi-Head Self-Attention, che permettono al modello di considerare simultaneamente diversi aspetti delle relazioni tra gli elementi della sequenza di input, di reti feed-forward, di connesioni residuali e di layer normalization.

3.4 Elaborazione dei dati e Addestramento con MoViNet

L'algoritmo utilizzato presentato in due funzioni principali progettate per estrarre e formattare fotogrammi da un file video.

La prima funzione, denominata *"format_frames"*, si occupa della formattazione dei singoli fotogrammi estratti. Essa converte l'immagine dal video in un formato di tipo *float32*, utile per l'elaborazione successiva. Inoltre, ridimensiona e aggiunge padding all'immagine per ottenere una dimensione specificata, garantendo così che tutti i fotogrammi abbiano le stesse dimensioni indipendentemente dalla risoluzione originale del video.

La seconda funzione, *"frames_from_video_file"*, gestisce il processo di estrazione dei fotogrammi da un video. Dopo aver aperto il file video utilizzando *OpenCV*, calcola il numero totale di fotogrammi presenti nel video. Se il video contiene un numero sufficiente di fotogrammi, estrae un fotogramma ogni 15, altrimenti calcola un intervallo di estrazione basato sul numero totale di fotogrammi disponibili e il numero di fotogrammi richiesti. Questa funzione determina un punto di partenza casuale per l'estrazione dei fotogrammi, garantendo che il processo di estrazione sia distribuito uniformemente lungo la durata del video.

Dopo aver settato la posizione iniziale, la funzione inizia a leggere i fotogrammi dal video. Il primo fotogramma viene sempre letto e formattato, mentre i successivi vengono estratti a intervalli regolari definiti dall'intervallo di estrazione calcolato. Se la lettura di un fotogramma non riesce, la funzione sostituisce il fotogramma mancante con un array di zeri delle stesse dimensioni. Alla fine del processo, l'insieme di fotogrammi estratti viene convertito in un array *NumPy*, con l'ordine dei canali dei colori modificato per adattarsi al formato richiesto.

Questo approccio garantisce un metodo efficiente per estrarre e uniformare fotogrammi da un video, permettendo analisi successive coerenti e standardizzate indipendentemente dalle caratteristiche del video di origine. Per ogni video del dataset, nel nostro esperimento, sono stati presi 20 frame per ognuno. In particolare, per ogni video, viene scelto un frame random di partenza e, da questo, un frame ogni 15, fino a raggruppare 20 frame per ogni video. Di seguito è riportato il codice per l'elaborazione dei frame di ogni video.

Per l'addestramento è stato utilizzato un approccio di transfer learning tramite un modello MoViNet-A0 pre-addestrato sul dataset Kinetics-600. Dopodiché, sono stati congelati gli strati del backbone ed è stato costruito un nuovo classificatore in cima alla rete con il numero di classi necessarie per il nostro esperimento. Una volta preparato il modello, è stato eseguito un addestramento di 30 epoche per effettuare fine-tuning sul

dataset trattato nel precedente capitolo. Il set di training è stato suddiviso in batch da 16 video ciascuno.

3.5 Elaborazione dei dati e Addestramento con Transformers

Per quanto riguarda il modello a trasformatori, sono state utilizzate le stesse funzioni per estrarre i frame dai video. Dopo aver estratto i nostri campioni da ogni video, è stato provveduto a trasformare queste immagini in feature map.

Una feature map, o mappa delle caratteristiche, è un concetto chiave nelle reti neurali convoluzionali (CNN). Essa rappresenta l'output di un livello convoluzionale dopo l'applicazione di filtri (o kernel) a un'immagine di input. In pratica, una feature map è il risultato dell'operazione di convoluzione, dove ogni filtro mette in evidenza specifici pattern o caratteristiche dell'immagine, come bordi, texture o altre particolarità visive.

Ogni feature map cattura specifici aspetti dell'immagine, come contorni, angoli, o pattern più complessi nelle reti più profonde. La convoluzione e il pooling (sottocampionamento) riducono la dimensione spaziale dell'immagine, mantenendo comunque le informazioni rilevanti. Il numero di feature map generate in un livello convoluzionale corrisponde al numero di filtri applicati, aumentando la profondità (dimensione del canale) dell'immagine rappresentata.

I vantaggi dell'utilizzo delle feature map sono molteplici. In primo luogo, le operazioni di convoluzione e pooling riducono le dimensioni spaziali delle immagini, permettendo alla rete di gestire input di grandi dimensioni in modo più efficiente. Inoltre, ogni filtro è addestrato per rilevare specifici pattern nelle immagini. Le feature map risultanti enfatizzano queste caratteristiche rilevanti, facilitando la successiva classificazione o rilevazione.

Le reti convoluzionali, attraverso l'uso delle feature map, sono intrinsecamente robuste alle traslazioni e ad altre trasformazioni locali dell'immagine, migliorando la generalizzazione del modello.

All'inizio è stata seguita una precisa strategia: prima sono state estratte le feature map con la rete neurale DenseNet preaddestrata su ImageNet; ma il modello non dava risultati. Ciò è probabilmente dovuto dal fatto che questa rete non riesce ad estrarre bene le caratteristiche utili. È stata quindi presa la nostra rete preaddestrata MoViNet originale, che si occupava di classificare le azioni nel dataset *Kinetics600*, rimuovendone il layer di classificazione. Così facendo, otteniamo una rete che in input prende delle immagini ed in output e, dopo varie convoluzioni, produce una feature map. Grazie a questa strategia, il modello a trasformatori ha raggiunto ottimi risultati di classificazione video per entrambe le versioni del dataset.

4 Risultati e Discussioni

In questa sezione verranno discussi i risultati raggiunti.

4.1 Risultati della rete MoViNet

Di seguito verranno riportate le metriche ottenute con l'addestramento del modello MoViNet per entrambe le suddivisioni di classi scelte.

4.1.1 Classi di pericolosità

Nella Tabella 3 sono riportate le metriche risultanti dal modello addestrato utilizzando classi che fanno riferimento al livello di pericolosità dell'azione condotta nel video.

	Classe 0	Classe 1	Classe 2	
Precision	0.85	0.86	1.00	
Recall	0.94	0.95	0.82	
F1-score	0.89	0.90	0.90	
Accuracy				0.90

Table 3 Risultati di Precision, Recall, F1-score e Accuracy per ciascuna classe di pericolosità con MoViNet.

Il modello ha ottenuto buoni risultati nel classificare correttamente i video nel set di test, raggiungendo un'accuratezza del 90%. Per quanto riguarda la precisione, notiamo che tutte le predizioni fatte per la classe di pericolo alto sono state corrette; per le classi 0 e 1 il modello ha ottenuto valori di precisione 0.85 e 0.86 rispettivamente. Dai valori di recall possiamo vedere come la rete abbia identifiato correttamente il 94% e il 95% degli esempi appartenenti alla classe 0 e 1 rispettivamente, scendendo all'82% degli esempi della classe 2. Infine, dai valori di F1-score vediamo che generalmente è efficace nel classificare gli esempi del test set, registrando valori intorno a 0.90 per tutte le classi. Per una visione dei risultati di classificazione più dettagliata, in Figura 1 è riportata la matrice di confusione.

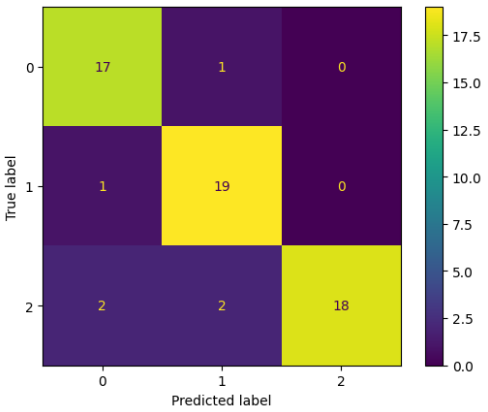

Fig. 1 Matrice di confusione classi di pericolosità ottenuta con MoViNet.

4.1.2 Classi sul tipo di arma

Nella Tabella 4 sono riportate le metriche risultanti dal modello addestrato utilizzando classi che fanno riferimento al tipo di arma mostrata nel video.

	Classe 0	Classe 1	Classe 2	
Precision	1.00	0.90	0.87	
Recall	0.89	0.90	0.95	
F1-score	0.94	0.95	0.91	
Accuracy				0.92

Table 4 Risultati di Precision, Recall, F1-score e Accuracy per ciascuna classe di arma con MoViNet.

Anche in questo esperimento abbiamo ottenuto risultati globalmente buoni, raggiungendo un'accuratezza del 92%. Più nel dettaglio, notiamo che tutte le predizioni fatte dal modello per la classe 0 (nessuna arma) sono corrette, ottenendo un valore di precision di 1.0. Qualche falso positivo, invece, ha fatto scendere la precisione a 0.87 per la classe 2. Osservando i valori di recall, possiamo vedere che il 95% degli esempi della classe 2 e circa il 90% degli esempi delle altre classi sono stati classificati correttamente. Riguardo all'F1-score, il modello ha registrato riusultati tra 0.91 e 0.95 per tutte le classi di armi. In Figura 2 è riportata la matrice di confusione relativa a questi risultati.

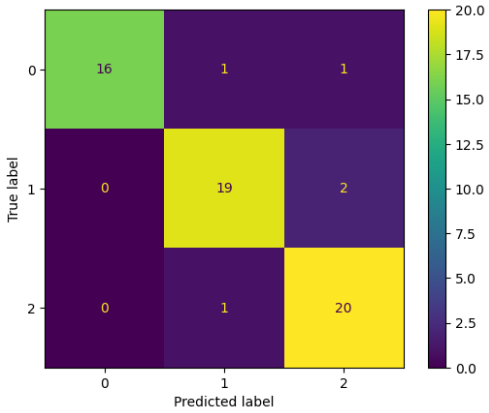

Fig. 2 Matrice di confusione classi di arma ottenuta con MoViNet.

4.2 Risultati della rete Transformers

Di seguito sono riportate le metriche ottenute con l'addestramento del modello Transformers per entrambe le suddivisioni di classi scelte.

4.2.1 Classi di pericolosità

Nella tabella 5 sono riportate le metriche risultanti dal modello addestrato utilizzando classi che fanno riferimento al livello di preicolosità mostrata nel video.

	Classe 0	Classe 1	Classe 2	
Precision	0.94	0.84	0.87	
Recall	0.94	0.80	0.91	
F1-score	0.94	0.82	0.89	
Accuracy				0.88

Table 5 Risultati di Precision, Recall, F1-score e Accuracy per ciascun grado di pericolosità.

Il modello mostra una forte capacità di classificazione per la maggior parte delle classi, con particolare efficacia per la classe 0, come indicato dai suoi elevati valori di precisione, recall e F1-score. La classe 1 ha un po' più di difficoltà, con valori di precisione e recall leggermente inferiori rispetto alle altre classi, suggerendo che potrebbero esserci più errori di classificazione in questa categoria.

Di seguito, figura 3, viene riportata la matrice di confusione.

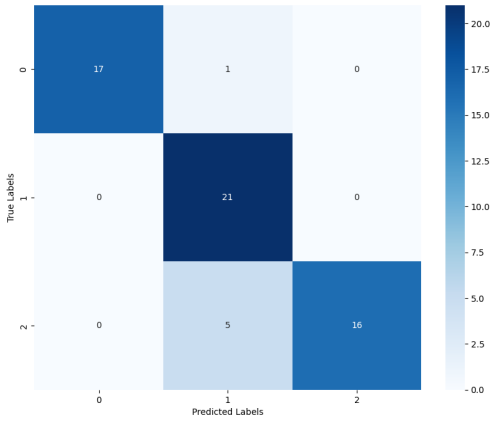

Fig. 3 Matrice di confusione modello basato su transformers per il classificatore di rischio.

4.2.2 Classi sul tipo di arma

Nella Tabella 6 sono riportate le metriche risultanti dal modello addestrato utilizzando classi che fanno riferimento al tipo di arma mostrata nel video.

L'accuratezza del test è aumentata, e le metriche di precisione, recall e F1-score sono migliorate per tutte le classi. In particolare, la classe 0 ha raggiunto una precisione e un recall perfetti, e le classi 1 e 2 mostrano una migliore performance bilanciata. Di seguito, figura 4, viene riportata la matrice di confusione.

	Classe 0	Classe 1	Classe 2	
Precision	1.00	0.91	0.95	
Recall	1.00	0.95	0.90	
F1-score	1.00	0.93	0.93	
Accuracy				0.95

Table 6 Risultati di Precision, Recall, F1-score e Accuracy per ciascuna classe di arma.

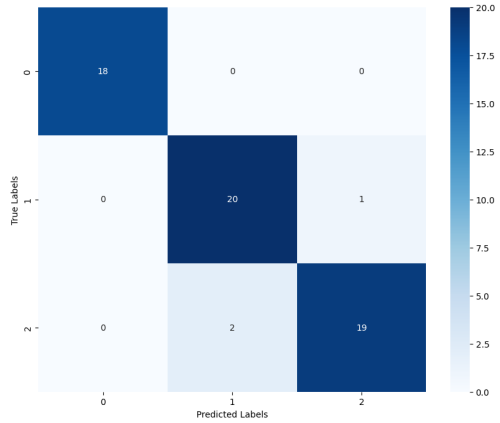

Fig. 4 Matrice di confusione modello basato su transformers per il riconoscimento dell'arma da fuoco.

4.3 Impatto ambientale per l'addestramento di MoViNet

L'addestramento dei modelli è stato monitorato utilizzando il tool CodeCarbon, che ha fornito i seguenti dati sui consumi energetici:

- Consumo Energetico RAM Totale: 0.014632 kWh.
- Consumo Energetico GPU Totale: 0.101851 kWh.
- Consumo Energetico CPU Totale: 0.130940 kWh.
- Consumo Totale di Elettricità: 0.247422 kWh.
- Emissioni Totali di CO2: 0.104368 kg CO2eq.

Per contestualizzare l'impatto ambientale dell'addestramento dei modelli di deep learning, possiamo confrontare i consumi energetici e le emissioni di CO2 con altre attività comuni.

La guida di un'auto a benzina, ad esempio, produce circa 2.31 kg di CO2 per ogni litro di carburante consumato. Pertanto, le emissioni di CO2 generate durante l'addestramento dei modelli sono equivalenti a quelle prodotte dalla combustione di circa 45 millilitri di benzina.

Un altro confronto utile riguarda l'uso di un computer portatile. In media, un laptop consuma circa 50 Wh di energia elettrica per ora. L'addestramento dei modelli ha consumato 0.247422 kWh, un valore che corrisponde all'uso di un computer portatile per circa 4.95 ore.

Infine, consideriamo il consumo energetico di una lampadina LED da 10 W, che consuma 10 Wh di energia elettrica per ora. L'energia consumata durante l'addestramento dei modelli è equivalente all'uso di una lampadina LED per circa 24.74 ore.

Questi confronti aiutano a mettere in prospettiva l'impatto energetico e ambientale dell'addestramento dei modelli di deep learning rispetto ad attività quotidiane più familiari.

4.4 Impatto ambientale per l'addestramento della rete Transformers

L'addestramento del modello è stato monitorato utilizzando il tool CodeCarbon, che ha fornito i seguenti dati sui consumi energetici:

- Consumo Energetico RAM Totale: 0.001706 kWh
- Consumo Energetico GPU Totale: 0.010548 kWh
- Consumo Energetico CPU Totale: 0.015264 kWh
- Consumo Totale di Elettricità: 0.027518 kWh
- Emissioni Totali di CO2: 0.006931 kg

Continuando i confronti precedenti, le emissioni di CO2 generate durante l'addestramento dei modelli sono equivalenti a quelle prodotte dalla combustione di circa 2.84 millilitri di benzina; corrisponde anche all'uso di un computer portatile per circa 0.55 ore ed infine, considerando la lampadina LED da 10 W, sono circa 2.75 ore di funzionamento.

Come è facile notare rispetto all'esempio precedente, l'addestramento del modello a trasformatori si è dimostrato molto più efficiente in termini energetici e di tempo.

4.5 Conclusioni e sviluppi futuri

Il presente lavoro ha esplorato due approcci distinti per il riconoscimento delle azioni legate all'uso di armi da fuoco nei video: l'architettura MoViNet e una rete basata su trasformatori. I risultati sperimentali hanno dimostrato che MoViNet, particolarmente nella sua versione base multi-clip, ha ottenuto buone prestazioni in entrambi i task di classificazione, nonostante sia stata testata la sua versione più leggera. MoViNet ha dimostrato una notevole capacità di catturare le dinamiche temporali e spaziali dei video, ottenendo valori di precisione, richiamo e F1-score spesso maggiori di 0.90 per entrambe le suddivisoni di classi considerate. Nel nostro caso, i tempi di addestramento si sono rivelati molto più lunghi a parità di hardware rispetto al modello basato su trasformatori. Tuttavia, durante i test, sono stati registrati tempi di inferenza piuttosto bassi, richiedendo in media 70 ms per video.

Per quanto riguarda la rete basata su trasformatori, abbiamo avuto dei dati altrettanto validi. Il vero vantaggio di questa architettura, tuttavia, è stato il minor consumo energetico ed il minor tempo di addestramento. Ciò è stato reso possibile grazie alla previa estrazione delle feature map, che ha accelerato notevolmente la fase di training. Ciononostante, i tempi di inferenza sono maggiori rispetto agli esperimenti precedenti, registrando tempi di inferenza che oscillano tra 1 e 2 secondi per video.

Un'interessante direzione per i futuri sviluppi riguarda l'implementazione di questi modelli su dispositivi edge per applicazioni di sicurezza e sorveglianza in tempo reale, soprattuto per quanto riguarda il modello MoViNet. I dispositivi edge, come le telecamere intelligenti e i droni, richiedono modelli di deep learning che siano non solo precisi ma anche efficienti in termini di risorse computazionali e consumo energetico. Nello specifico, è auspicabile intervenire su:

1. Ottimizzazione del Modello: Ridurre ulteriormente la complessità dei modelli mantenendo l'accuratezza può rendere l'implementazione su dispositivi edge più pratica. Tecniche di quantizzazione e pruning possono essere utilizzate per questo scopo.
2. Inferenza in Tempo Reale: Garantire che le due reti possano eseguire inferenze in tempo reale su hardware limitato è fondamentale per applicazioni di sorveglianza. L'uso di acceleratori hardware dedicati come TPU (Tensor Processing Units) o NPU (Neural Processing Units) può migliorare significativamente le prestazioni.
3. Robustezza e Sicurezza: Sviluppare algoritmi per garantire la robustezza del modello contro attacchi avversariali e condizioni di rete instabili è cruciale per la sicurezza delle applicazioni edge.
4. Validazione sul Campo: Eseguire test e validazioni in ambienti reali per garantire che le due reti sviluppate funzionino efficacemente in scenari pratici di sorveglianza e sicurezza.

References

[1] Kondratyuk, D., Yuan, L., Li, Y., Zhang, L., Tan, M., Brown, M., Gong, B.: Movinets: Mobile video networks for efficient video recognition. Data in Brief (2021)

[2] Ruiz-Santaquiteria, J., Muñoz, J.D., Maigler, F.J., Deniz, O., Bueno, G.: Firearm-related action recognition and object detection dataset for video surveillance systems. Data in Brief **52**, 110030 (2024) https://doi.org/10.1016/j.dib.2024.110030

[3] Lim, J., Al Jobayer, M.I., Baskaran, V.M., Lim, J.M., Wong, K., See, J.: Gun detection in surveillance videos using deep neural networks, 1998–2002 (2019) https://doi.org/10.1109/APSIPAASC47483.2019.9023182

[4] Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H.: M2det: A single-shot object detector based on multi-level feature pyramid network. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 9259–9266 (2019) https://doi.org/10.1609/aaai.v33i01.33019259

[5] Kanehisa, R., Neto, A.: Firearm detection using convolutional neural networks, 707–714 (2019) https://doi.org/10.5220/0007397707070714

[6] Olmos, R., Tabik, S., Herrera, F.: Automatic handgun detection alarm in videos using deep learning. Neurocomputing **275**, 66–72 (2018) https://doi.org/10.1016/j.neucom.2017.05.012