Ottimizzazione in ambito Edge di modelli per Violence Detection

Luca Bellante[1†] and Agnese Bruglia[2†]

[1]Dipartimento di Ingegneria dell'Informazione, UNIVPM, Via Brecce Bianche 12, Ancona, 60131, Italia.
[2]Dipartimento di Ingegneria dell'Informazione, UNIVPM, Via Brecce Bianche 12, Ancona, 60131, Italia.

Contributing authors: s1118814@studenti.univpm.it;
s1120034@studenti.univpm.it;
[†]Gli autori hanno contribuito al lavoro in egual misura.

Abstract

Il seguente lavoro ha l'obiettivo di approfondire le tecniche e i meccanismi di rilevamento della violenza. Questo studio mira a trasferire tali tecniche in ambito edge, con l'intento di sviluppare soluzioni che possano essere installate e utilizzate in modo efficace ed efficiente vicino a dove immagini e video vengono registrate anziché in cloud, riducendo così la latenza, migliorando la reattività del sistema e permettendo di realizzare soluzioni rispettose della privacy. In questo contesto, l'implementazione di algoritmi avanzati di violence detection direttamente sulle videocamere di sorveglianza rappresenta una significativa innovazione, potenzialmente capace di aumentare la sicurezza pubblica e privata. Questo lavoro si propone, quindi, di contribuire alla creazione di un sistema di sorveglianza più intelligente e reattivo, in grado di rilevare e segnalare tempestivamente situazioni di pericolo, migliorando così la capacità di intervento e prevenzione delle forze dell'ordine e delle organizzazioni di sicurezza.

Keywords: Violence detection, CCTV surveillance, Deep Learning, Edge AI

1 Introduzione

La violenza è un problema globale che affligge molte società causando danni fisici, psicologici ed economici; perciò identificare in maniera efficace episodi di violenza e intervenire tempestivamente è cruciale per prevenire danni a persone e/o cose. In Italia tra il 2019 e il 2022 c'è stato un notevole aumento di atti violenti in luoghi pubblici da parte di miniori [1]. Questa situazione rappresenta una sfida etica e morale, affrontata anche tramite il controllo e la supervisione del territorio con le telecamere a circuito chiuso dei Comuni.
Tradizionalmente, i sistemi di sorveglianza basati su telecamere richiedono una supervisione umana costante per individuare atti di violenza, un compito oneroso e soprattutto soggetto a errori umani. Negli ultimi anni, quindi, l'avanzamento delle tecnologie di Computer Vision ha aperto nuove possibilità grazie al riconoscimento automatico di comportamenti violenti in contesti pubblici e privati, anche se, l'implementazione di tali sistemi su larga scala presenta sfide significative in termini di latenza, privacy e costi di comunicazione.
In questo contesto, l'edge computing emerge come una soluzione innovativa, in grado di portare il calcolo vicino alla fonte dei dati: questi sistemi, infatti, possono elaborare le informazioni in loco, riducendo la latenza e migliorando la privacy, poiché i dati sensibili non devono essere trasmessi a server remoti. Rappresentano perciò un passo avanti sia nella sicurezza pubblica che privata, sebbene mantengano gli ostacoli di una applicazione su sistemi edge, ovvero limitazioni di potenza computazionale e memoria.
È necessario quindi utilizzare un modello di rete neurale leggero e che sia in grado di ottenere buone prestazioni in questa tipologia di sistemi. Per questo motivo, in questo elaborato viene presentato uno studio delle migliori tecniche di ottimizzazione che consentono a modelli per il riconoscimento di violenza un buon livello di accuratezza e tempo di elaborazione nonostante la riduzione delle dimensioni.

2 Analisi dello stato dell'arte

Analizzando lo stato dell'arte attuale non strettamente legato all'ambito edge e relativo al problema di violence detection si trovano innumerevoli e diverse tipologie di soluzioni esistenti, perciò di seguito se ne propone qualcuna ritenuta più rilevante.
La prima soluzione analizzata [2] propone tre diversi approcci:

- **C3D**[1] **+ SVM**[2]: viene utilizzata una C3D pre-addestrata su ImageNet per estrarre features con le quali addestrare il modello *SVM*. La funzione di attivazione finale è sigmoidale, in quanto il problema di classificazione è binario;
- **C3D + Fully Connected Layers**: gli strati finali densamente connessi sono addestrati a partire da un'estrazione di features della rete C3D e la funzione di attiviazione finale è sigmoidale;
- **ConvLSTM + Fully Connected Layers**: a differenza delle soluzioni precedenti, sia la *ConvLSTM* che i *Fully-Connected-Layers* sono addestrati da zero.

Di questi tre modelli, in seguito a test effettuati su tre dataset [3] [4] [5], il modello formato da C3D + SVM è risultato essere il migliore in termini di stabilità di accuratezza.

[1] Il modello C3D (Convolutional 3D) è una rete neurale convoluzionale tridimensionale progettata per analizzare video e sequenze temporali, catturando informazioni spaziali e temporali per applicazioni come il riconoscimento di azioni, l'analisi medica e la videosorveglianza

[2] Il Support Vector Machine (SVM) è un algoritmo di apprendimento supervisionato utilizzato per classificazione e regressione, che separa i dati trovando l'iperpiano ottimale in uno spazio multidimensionale.

Un'altra tipologia di architettura possibile [6] è formata da convoluzioni 2D e layer neurali ricorrenti, o "RNN", cioè *LSTM* o *GRU*, con lo scopo di estrarre caratteristiche spazio-temporali al fine di classificare in maniera efficace l'azione presente nei frame.

Per quanto riguarda, invece, l'ambito edge, gli autori dell'articolo [7] offrono una valida alternativa ai modelli con RNN: sfruttano una rete chiamata *ST-TCN* in cui vengono effettuate convoluzioni spazio-temporali al fine di estrarre, appunto, features sia spaziali che temporali. Qui, Una volta ridotte la dimensionalità dei frame per questioni di efficienza e adattabilità alla bootleneck, le mappe di output attraversano dei "*blocchi di attenzione*", che rappresentano una modifica del bootleneck della *resNet*. Alla fine l'output viene decompresso, mandando in uscita la probabilità che un determinato frame o gruppi di frame rappresentino una scena di violenza. Hanno così dimostrato come tale architettura sia relativamente efficiente e scalabile, anche se ha ancora bisogno di essere migliorata prima di essere in grado di gestire video real time.

2.1 Architettura con MobileNetV2

Di seguito si tratta in maniera sintetica ed efficace quanto fatto dagli autori dell'articolo [7]. Tale attenzione è dettata dal fatto che le strutture qui proposte sono state scelte come base per il lavoro svolto in questo elaborato, cioè come rampa di lancio per ottimizzazioni e trasporto in ambito edge.

Le architetture proposte sono due:

- **MobileNetV2 + BiLSTM + Layer denso**;
- **MobileNetV2 + ConvLSTM + Layer denso**.

Si passa ora ad analizzare nel dettaglio le architetture proposte.
La rete *MobileNetV2* riprende il principale punto di forza della precedente *MobileNetV1*, ovvero le convoluzioni separabili in profondità, aggiungendo però ulteriori caratteristiche con lo scopo di aumentare l'efficienza e l'efficacia delle features estratte:

- **Linear Bottlenecks**: l'aggiunta di un "Collo di Bottiglia lineare" ha lo scopo di ridurre le dimensioni dell'immagine di input, favorendo calcoli più veloci e aumentando così l'efficienza complessiva del modello. Inoltre, i layer del Bottleneks sono "*lineari*" in quanto i creatori del modello [8] hanno verificato che layer non lineari distruggono troppe informazioni, risultando in un abbassamento dell'efficacia complessiva del modello stesso;
- **Blocchi residuali invertiti**: per "blocchi residuali" si intende un collegamento diretto, o cortocircuito, all'interno di un layer convoluzionale, utile per migliorare la retro-propagazione del gradiente e che, solitamente, avviene tra una compressione e una successiva decompressione. L'intuizione degli autori di [8] è stata quella di effettuare un cortocircuito tra una decompressione ed una compressione, intuizione che ha portato a una retropropagazione più efficace del gradiente.

Il layer *BiLSTM*, invece, è composto da due sequenze di layer LSTM (Long Short-Term Memory), ciascuna con neuroni e pesi indipendenti. Le informazioni sono analizzate simultaneamente in entrambe le direzioni, avanti e indietro, e ogni volta che l'informazione passa da un neurone all'altro la cella di memoria relativa al neurone nel layer LSTM corrispondente viene aggiornata. L'output finale del layer è il risultato di una combinazione di informazioni estratte da ciascun neurone sia della LSTM in avanti che dalla LSTM all'indietro.

D'altra parte, la *ConvLSTM* è una LSTM in cui le moltiplicazioni interne alle matrici vengono sostituite con operazioni di convoluzione, rendendo il modello complessivo più efficiente e mediamente poco pesante.

Infine, trattandosi di un problema di classificazione binario, gli autori di [7] per entrambe le tipologie di architetture mezionate, hanno fatto convogliare gli output dei layer *ConvLSTM* e *BiLSTM* all'interno di strati densamente connessi, la cui funzione di attivazione finale è la sigmoide.

3 Materiali e Metodi

3.1 Dataset

Il dataset utilizzato per training, validation e testing è "AirtLab Dataset" [2] [3]. Questo dataset presenta diversi vantaggi, come il fatto di essere stato realizzato con il fine di prevenire falsi positivi, includendo nei campioni video rappresentanti azioni come strette di mano e abbracci che potrebbero essere scambiati come comportamenti violenti. Altri dataset sono stati considerati come base di addestramento per l'architettura proposta, come "Real Life Violence Situations Dataset" [9] e "Not only Look, but also Listen: Learning Multimodal Violence Detection under Weak Supervision" [10], entrambi però presentavano delle problematiche: il primo include video di natura troppo varia, da filmati di sicurezza di telecamere con qualità estremamente bassa a video registrati da telefoni cellulare con inquadratura tutt'altro che fissa, situazioni che non combaciano con il dominio di applicazione del progetto sviluppato; mentre il secondo presenta la principale difficoltà di includere una percentuale molto elevata di video violenti in cui la situazione violenta non è isolata, cioè la maggior parte di tale video presenta in realtà una scena non violenta, risultando quindi in un dataset non ottimale. Maggiori dettagli sono presenti nelle tabelle: 2 e 1.

Numero totale di frame	Media FPS	Durata media video(s)
Violence		
159782	29.57	5.40
Non Violence		
127587	25.01	5.10

Table 1: Informazioni estratte dal dataset *Real Life Violence Situations Dataset* [9].

AirtLab Dataset, invece, include 350 video Full HD con una media di 30 frame al secondo ed è organizzato in maniera efficiente in due cartelle principali, "violent", con 230 video, e "non-violent", con 120 video, a loro volta suddivise in due sotto-cartelle, "cam1" e "cam2", che contengono gli stessi video ripresi da due prospettive diverse. La decisione di utilizzare questo dataset, oltre a essere dovuta ai vantaggi sopra elencati, è legata alla volontà di avere una continuità nel lavoro rispetto allo stato di partenza [2] con il fine di presentare un confronto lineare con questo.

Inoltre, a partire da "AirtLab Dataset" sono stati realizzati i set su cui si sono fatti fine-tuning e inferenza, di cui si parla nella sezione 3.3. In particolare, il set destinato al fine-tuning è stato costituito con alcuni video, per un totale di 90s, presi dal dataset principale su cui è stata applicata la tecnica di data augmentation aggiungendo sfocatura e riducendo la risoluzione dei

Numero totale di frame	Media FPS	Durata media video(s)	Numero totale video	Percentuale di filmati di durata minore o uguale a 6 secondi
Train				
16318199	24	172.13	3950	**0.15%**
Test				
2335801	24	121.66	800	**0.12%**

Table 2: Informazioni estratte dal dataset *Not only Look, but also Listen: Learning Multimodal Violence Detection under Weak Supervision* [10].

frame del 30%; mentre il set destinato all'inferenza include solamente alcuni video selezionati dal dataset di partenza per un totale di 102s. In generale, i video appartenenti a ogni set vengono preprocessati prima di essere mandati alla rete neurale in fase sia di addestramento, sia di fine-tuning e sia di inferenza: ogni video viene diviso in batch da 16 frames l'uno.
Infine, si fa notare che il set di addestramento e i set per il fine-tuning, ma soprattutto, per l'inferenza includono gli stessi video per diverse ragioni: una è che esistono pochi dataset online che includano effettivamente video con inquadratura fissa, come sarebbe quella di una videocamera di sorveglianza, e che abbiano una risoluzione accettabile, ma la ragione principale è che, per quanto concerne lo scopo del lavoro, è necessario non tanto testare l'efficacia generale del modello finale, ma il variare di questa al variare del peso e del numero di parametri del modello a seguito dell'applicazione delle tecniche di ottimizzazione.

3.2 Architettura proposta

La soluzione proposta in questo elaborato è stata pensata a partire da quella discussa a sezione 2.1 sostituendo la rete MobileNetV2 con la MobileNetv3 Small. Questa soluzione prevede la proposta di due architetture differenti:

- **MobileNetV3 Small + BiLSTM + Layer denso**;
- **MobileNetV3 Small + ConvLSTM + Layer denso**.

La scelta di sostituire La MobileNetV2 con la MobileNetV3 Small è stata dettata dalla natura di quest'ultima, che è una rete convoluzionale progettata per dispositivi mobili e ottimizzata per avere buone prestazioni con un numero ridotto di parametri e operazioni, ottenendo perciò, su sistemi edge, prestazioni migliori nonostante le dimensioni ridotte non solo rispetto alla MobileNetV2 [11], ma anche rispetto a modelli di reti più recenti come la MobileNetV3 Large. Si è quindi scelto di utilizzare la MobileNetV3 Small pre-addestrata con i pesi di Imagenet e avvolta da un layer TimeDistributed, che le permette di elaborare dati temporali, come backbone dell'architettura finale.
In particolare, le tabelle 3 e 4 rappresentano rispettivamente le architetture delle soluzioni con la ConvLSTM e con la BiLSTM: si può notare che in entrambe è presente un primo layer "TimeDistributed", che rappresenta la backbone dell'architettura, cioè la MobileNetV3 Small, seguito nel primo caso da un layer "ConvLSTM2D" e nel secondo caso da un layer "Bidirectional", LSTM bidirezionale, per poi terminare con due layer "Dense" fully connected. Come per le reti di cui si è parlato nella sezione 2.1, per entrambe le architetture proposte, le fasi di training e testing sono state effettuate sul dataset AirtLab applicando una

stratified shuffle split cross-validation, ovvero è stata ripetuta una suddivisione casuale 80-20 per 5 volte, utilizzando l'80% dei dati come set di addestramento e il 20% come set di test, preservando la percentuale di campioni di ciascuna classe in ogni split.

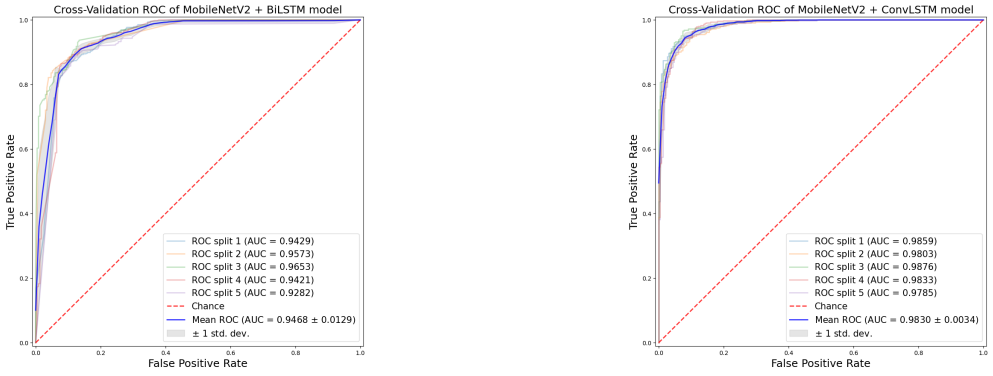
Layer	Output Shape	Param #
TimeDistributed	(None, 16, 7, 7, 576)	939120
ConvLSTM2D	(None, 5, 5, 64)	1474816
Flatten	(None, 1600)	0
Dropout	(None, 1600)	0
Dense	(None, 256)	409856
Dropout	(None, 256)	0
Dense	(None, 1)	257

Table 3: Dettagli della rete con la ConvLSTM

Layer	Output Shape	Param #
TimeDistributed	(None, 16, 7, 7, 576)	939120
TimeDistributed	(None, 16, 28224)	0
Bidirectional	(None, 256)	29033472
Dropout	(None, 256)	0
Dense	(None, 128)	32896
Dropout	(None, 128)	0
Dense	(None, 1)	129

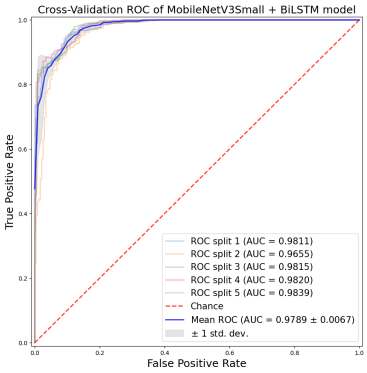
Table 4: Dettagli della rete con la BiLSTM

A proposito di stratified shuffle split cross-validation, le figure 1 e 2 rappresentano le curve ROC dei 5 split ottenute dalla fase di testing a conclusione dell'addestramento, rispettivamente delle architetture con MobileNetV2, BiLSTM (figura 1a) e ConvLSTM (figura 1b), e di quelle con la MobileNetV3 Small, BiLSTM (figura 2a) e ConvLSTM (figura 2b).

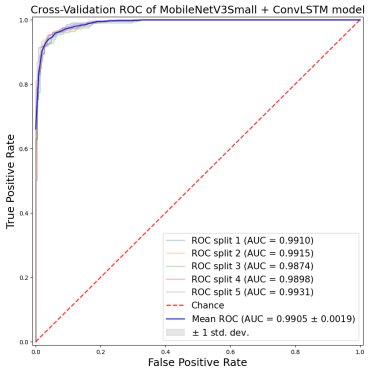


(a) BiLSTM (b) ConvLSTM

Fig. 1: ROC delle reti con MobileNetV2



(a) BiLSTM



(b) ConvLSTM

Fig. 2: ROC delle reti con MobileNetV3

Da queste figure si può già notare che i modelli con la MobileNetV3 Small sembrerebbero essere migliori rispetto a quelli con la MobileNetV2 e, ancora, che i modelli con la ConvLSTM sembrerebbero essere migliori di quelli con la BiLSTM.
Nella tabella 5, invece, sono riportati i risultati dell'inferenza, effettuata sul set dedicato come esposto nella sezione 3.1, sia delle architetture con la MobileNetV2 sia di quelle con la MobileNetV3. Confrontandole, si nota subito che le reti con la MobileNetV3 Small come backbone hanno un grado di accuratezza maggiore: infatti, la rete con la BiLSTM ha un'accuratezza del 90.6% e quella con la ConvLSTM del 95%, mentre le reti con la MobileNetV2 come backbone hanno, rispettivamente, un'accuratezza del 48.6% e del 65.7%

Modello	Accuratezza (%)	AUC(%)	Peso modello (MB)	Tempo inferenza medio batch (s)	Consumo energetico medio (Wh)
V2+ConvLSTM	65.7	72	49	0.99	3.3
V2+BiLSTM	48.6	26	746	1.71	5
V3Small+ConvLSTM	95	99.4	25.5	0.43	1.8
V3Small+BiLSTM	90.6	97.84	336.6	0.7	2.2

Table 5: Risultati delle reti con MobileNetV2 e MobileNetV3 Small

Inoltre, ricordando che l'obiettivo del lavoro svolto è di realizzare un modello leggero che possa avere buone prestazioni su sistemi edge nonostante le dimensioni ridotte, è degno di nota il fatto che passando da un'architettura che usa la MobileNetV2 come backbone a una che usa la MobileNetV3 Small si sia già riusciti a ridurre le dimensioni dei modelli di una buona percentuale: per la rete con la BiLSTM si passa da una dimensione di 745.9 MB a 336.6 MB e per la rete con la ConvLSTM si passa da 49.1 MB a 25.5 MB, risultando in una diminuzione di dimensione del 55% circa nel primo caso e del 48% circa nel secondo.
Da questi dati è quindi possibile astrarre che le architetture proposte in questo elaborato sono migliori rispetto a quelle di partenza. Inoltre, volgendo un secondo sguardo alla tabella 5, tra

le architetture con la MobileNetV3 Small si nota che la rete con la ConvLSTM sembrerebbe classificare le scene di violenza/non violenza meglio rispetto alla rete con la BiLSTM.

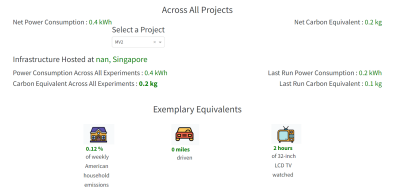
3.3 Metodi di ottimizzazione

Con il fine di garantire le migliori prestazioni in termini di tempo di inferenza e accuratezza in un sistema edge, si sono esplorate diverse possibili tecniche di ottimizzazione di modelli, principalmente post training, applicate sia alla rete con la ConvLSTM sia a quella con la BiLSTM: quantizzazione, clustering, pruning e una tecnica ibrida di pruning più quantizzazione. In particolare, la quantizzazione è una tecnica di riduzione della precisione numerica dei parametri di una rete neurale che può portare a benefici significativi in termini di riduzione della memoria necessaria per memorizzare il modello, della potenza computazionale richiesta per eseguire le operazioni e di riduzione del tempo di inferenza. Si sono esplorate, quindi, diverse tipologie di quantizzazione, ovvero la quantizzazione intera 8 bit con fallback a float, la quantizzazione intera 8 bit, la quantizzazione float a 16 bit e la quantizzazione aware training, unica tecnica che applica una quantizzazione intera 8 bit durante la fase di training e non post. Per ognuna di queste è stato necessario effettuare una ricalibrazione dei pesi post quantizzazione utilizzando i video del set di fine-tuning come spiegato nella sezione 3.1. Per quanto riguarda la quantizzazione Float16, durante i test di inferenza, è stato necessario utilizzare la GPU, questo poichè stando alla documentazione ufficiale Keras, che si può trovare al seguente link, il modello "dequantizzerà" a Float32 se eseguito dalla CPU.
Oltre alle tecniche sopra elencate, ne è stata testata una ulteriore che è ancora in fase sperimentale e che, infatti, ha dato problemi: la quantizzazione "16-bit activations with 8-bit weights", ovvero in cui le attivazioni sono quantizzate a interi a 16 bit, i pesi a interi a 8 bit e i bias a interi a 64 bit.
Il clustering, invece, serve a raggruppare insiemi di parametri in gruppi omogenei e, per ogni gruppo omogeneo, o "cluster", scegliere un centroide secondo algoritmi differenti. Anche questo approccio ha lo scopo di diminuire la complessità della rete neurale e la dimensione del modello. Più nel dettaglio, si sono provate varie combinazioni degli iperparametri numero di cluster e algoritmi di inizializzazione dei centroidi per cercare di trovare il modello che fosse il più efficace ed efficiente: i numeri di cluster provati sono stati 8, 16 e 32 e gli algoritmi di inizializzazione sono stati "linear", "random", "density based" e "kmeans++".
Per quanto riguarda il pruning, questa è una tecnica magnitude-based, cioè che annulla gradualmente una percentuale dei pesi della rete in base al loro valore assoluto durante il processo di addestramento in modo tale da ottenere sparsità nel modello: ciò rende il modello più facile da comprimere e, saltando gli zeri durante l'inferenza, permette di migliorare il tempo di inferenza. In particolare, sono state provate come percentuali di sparsità 40%, 50%, 60% e 80%.
Per entrambe le tecniche di clustering e pruning è stato necessario eseguire un fine-tuning di due epoche sui modelli ottimizzati utilizzando il dataset dedicato (sezione 3.1). Inoltre, sia per il clustering che per il pruning si è incontrato un ostacolo: non poter applicare l'ottimizzazione ai layer delle ConvLSTM e BiLSTM. Ciò è riconducibile al fatto che omogeneizzando o azzerando, rispettivamente, i pesi del modello, questo perderebbe la sua capacità di considerare informazioni a lungo termine.
Inoltre, per la tecnica di clustering, sebbene il seguente passaggio sia fondamentale se l'obiettivo è di ottenere un modello leggero, non è stato possibile eseguire il metodo "`tfmot.clustering.keras.strip_clustering()`", il cui scopo è liberare il

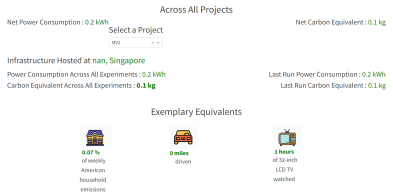
modello finale da informazioni superflue, dei “clustering wrappers”, necessarie per il fine-tuning ma non per l’inferenza, e ripristinare il modello iniziale con i pesi clusterizzati. In particolare è stata provata l’esecuzione con la CPU e GPU L4 della Nvidia messe a disposizione da Google Colab per diverse ore, ma è stato necessario interrompere l’esecuzione che sembrava essere finita in una situazione di loop. Il risultato è stato perciò un modello di dimensioni parecchio elevate rispetto a quelle potenziali, cioè di 558.4 MB per la rete con la BiLSTM e di 28.6 MB per la rete con la ConvLSTM, ma nonostante ciò si è deciso di testare comunque i modelli ottenuti.
Infine, tramite la tecnica ibrida di pruning più quantizzazione, si è cercato di combinare i vantaggi delle due per creare un modello ancora più leggero e che permettesse di avere tempi di inferenza ancora più brevi: sono state utilizzate le combinazioni di pruning al 50% e quantizzazione intera 8 bit con fallback a float e quantizzazione float a 16 bit.

4 Risultati e Discussione

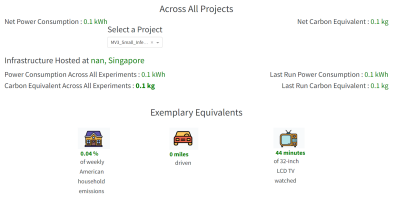
Durante lo svolgimento di questo lavoro è stato importante non lasciare da parte il concetto di consumo energetico: infatti la riduzione di dimensioni e di complessità dei modelli analizzati ha come fine anche di ridurre il più possibile non solo il consumo energetico stesso, ma anche la produzione di carbonio.



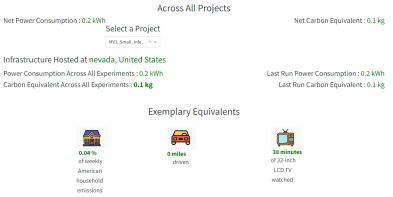
(a) MobileNetV2 + ConvLSTM



(b) MobileNetV2 + BiLSTM



(c) MobileNetV3 Small + ConvLSTM



(d) MobileNetV3 Small + BiLSTM

Fig. 3: *Carbon Footprint* dei modelli addestrati.

Per questo sono state utilizzate le API di CodeCarbon.io per misurare il valore di queste due variabili dovuto agli addestramenti delle reti neurali effettuati. In particolare, sono stati addestrati 5 modelli di reti neurali, cioè C3D+SVM, MobileNetV2+ConvLSTM, MobileNetV2+BiLSTM, MobileNetV3Small+ConvLSTM e MobileNetV3Small+BiLSTM, dei quali di seguito si riportano i consumi. Per quanto riguarda la produzione di carbonio,

la C3D+SVM ne ha prodotta una quantità talmente piccola da poter essere approssimata a 0 kg, la V2+ConvLSTM ne ha prodotti 0.2 kg e le altre 0.1 kg; inoltre, l'energia elettrica consumata durante gli adestramenti è stata di una quantità di nuovo approssimabile a 0 kWh per la rete C3D+SVM, di 0.4 kWh per la V2+ConvLSTM, di 0.1 kWh per la MobileNetV3Small+ConvLSTM e di 0.2 kWh per le altre. Tali consumi energetici sono riassunti in maniera chiara ed esplicita nella figura 3.
In aggiunta, si specifica che tutto il lavoro è stato eseguito utilizzando Google Colab con la CPU fornita dalla piattaforma. La scelta di utilizzare solamente la CPU è stata fatta con l'intenzione di ottenere risultati più realistici e applicabili, in quanto si considera che i modelli finali sono destinati a essere utilizzati in sistemi edge principalmente privi di GPU integrata. L'obiettivo era quindi quello di replicare il più possibile le condizioni operative reali. Solo in alcuni casi e per particolari tipi di ottimizzazioni è stato richiesto l'utilizzo della GPU-T4, ovvero, in tutte le tecniche di ottimizzazione che implementano la quantizzazione Float16.
Infine, si ricorda che i dati di partenza dei modelli della rete con la MobileNetV3 Small sono quelli riportati nella tabella 5 a sezione 3.2.

4.1 MobileNetV3 Small + ConvLSTM

Tecnica	Accuratezza (%)	AUC(%)	Peso modello (MB)	Tempo inferenza medio batch (s), *su 102 s di video in batch da 16 Frame*	Consumo energetico medio (Wh), *calcolato con 102 secondi di video*
Quantizzazione					
Intera a 8 bit con fallback Float	87.8	90.6	3.33	0.33	2
Intera a 8 bit	51	59	3.33	0.32	1.8
Float a 16 bit	95	99	6	0.40	1.8
Quantizzazione Aware Training	95	99	12	0.445	1.76
Clustering[3]					
8 Cluster e centroidi inizializzati a *densità*	96.1	99.6	28.6	0.42	1.23
8 Cluster e centroidi inizializzati *lineari*	95	99.5	28.6	0.41	1.6

[3] Il perchè del peso così elevato rispetto agli altri modelli, pur essendo una tecnica di ottimizzazione, è stato trattato al paragrafo 3.3

Tecnica	Accuratezza (%)	AUC(%)	Peso modello (MB)	Tempo inferenza medio batch (s), *su 102 s di video in batch da 16 Frame*	Consumo energetico medio (Wh), *calcolato con 102 secondi di video*
8 Cluster e centroidi inizializzati *random*	93.4	99.3	28.6	0.41	1.6
8 Cluster e centroidi inizializzati *Kmeans++*	95.6	99.6	28.6	0.41	1.6
16 Cluster e centroidi inizializzati a *densità*	96.1	99.6	28.6	0.59	1.9
16 Cluster e centroidi inizializzati *lineari*	95	99.5	28.6	0.42	0.86
16 Cluster e centroidi inizializzati *random*	95	99.46	28.6	0.4	1.2
16 Cluster e centroidi inizializzati *Kmeans++*	95	99.5	28.6	0.42	1.97
32 Cluster e centroidi inizializzati a *densità*	95.6	99.6	28.6	0.4	1.4
32 Cluster e centroidi inizializzati *lineari*	95.6	99.6	28.6	0.43	2.3
32 Cluster e centroidi inizializzati *random*	95	99.5	28.6	0.33	0.92

Tecnica	Accuratezza (%)	AUC(%)	Peso modello (MB)	Tempo inferenza medio batch (s), *su 102 s di video in batch da 16 Frame*	Consumo energetico medio (Wh), *calcolato con 102 secondi di video*
32 Cluster e centroidi inizializzati *Kmeans++*	95.6	99.55	28.6	0.45	1.2
Pruning					
Sparsità finale 40%	87.3	98	12	0.36	1.72
Sparsità finale 50%	94.5	98	12	0.38	2.3
Sparsità finale 60%	88.9	98	12	0.35	1.7
Sparsità finale 80%	84.5	98.3	12	0.36	1.9
Pruning + Quantizzazione					
Pruning 50% + QPT[4] int con fallback Float	80	85.5	3.3	0.35	1.8
Pruning 50% + QPT Float 16	83.98	97.87	6	0.3	2.3

Table 6: Risultati ottenuti con MobileNetV3 Small + ConvLSTM

In tabella 6 sono riportati i risultati dell'inferenza sui modelli ottimizzati a partire dalla rete *MobileNetV3Small+ConvLSTM*. E' doveroso e opportuno effettuare una valutazione incrociata tra i diversi criteri per le varie ottimizzazioni qui presenti. Occorre valutare non il valore in sè, bensì il delta relativo di una stessa metrica di giudizio tra più ottimizzazioni differenti, compreso il modello di partenza.
Per quanto riguarda le ottimizzazioni post e aware training, le soluzioni con il compromesso che si ritiene migliore tra accuratezza e "leggerezza" del modello sono:

1. **Quantizzazione Float a 16 bit**: il peso del modello è diminuito del **76.5%** rispetto al modello di partenza, mentre i valori di accuratezza e AUC sono rimasti pressochè stabili. Per quanto riguarda il tempo di inferenza medio, si è osservato un guadagno di circa il **7%**, mentre il consumo energetico è rimasto stabile. Tutto ciò è dovuto dal fatto che probabilmente i pesi del modello non hanno bisogno di una precisione *Float32* per generalizzare

[4]QPT sta per "Quantizzazione Post Training"

la realtà, quindi quindi passando a *FLoat16* si guadagna in termini di peso e, in parte, di efficienza;
2. **Quantizzazione intera 8 bit con fallback a Float**: il peso del modello è diminuito dell'**87%** rispetto al modello di partenza, mentre i valori di accuratezza e AUC sono diminuiti rispettivamente del **7%** e dell'**8.8%**, mentre il tempo di inferenza è migliorato del **25.6%** rispetto al modello di partenza.

Tra queste due tecniche, la quantizzazione Float16 ha un peso e una latenza superiore, rispettivamente del **10.5%** e **18.6%** rispetto al modello quantizzato Int con fallback a Float, ma a differenza di quest'ultimo mantiene accuratezza e valore AUC pari al modello di partenza non ottimizzato, con un'assorbimento in *Wh* di poco minore rispetto al modello Int con fallback Float.
Invece, per quanto riguarda la clusterizzazione, la trattazione è analoga rispetto a quanto detto e fatto per la quantizzazione. Purtroppo però, a causa dei problemi descritti nella sezione 3.3, il peso non sarà preso come riferimento per la selezione del miglior modello clusterizzato, e di conseguenza nemmeno il consumo energetico medio. Con riferimento ai limiti della clusterizzazione trattati al capitolo 3.3, osservando la figura 4, si comprende chiaramente che la clusterizzazione viene applicata solo a metà o poco meno del modello di partenza.

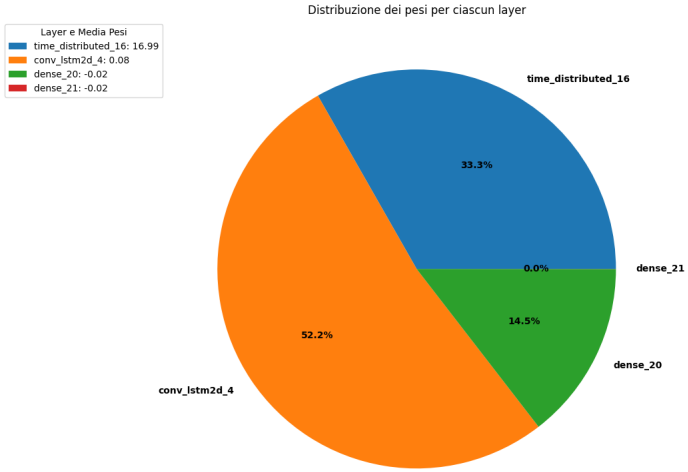
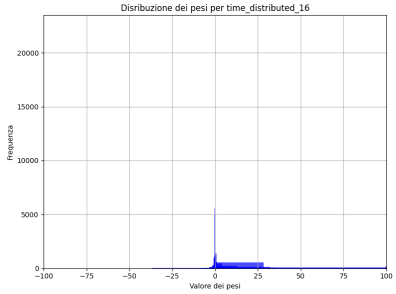


Fig. 4: Pesi nei vari layer del modello *MobileNetV3 Small + ConvLSTM*.

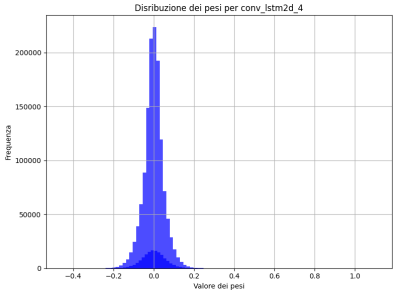
I modelli "migliori" sono:

1. **16 Cluster e centroidi lineari**: l'accuratezza è rimasta stabile, l'AUC è aumentata dello **0.1%** e il tempo di inferenza medio è diminuito del **2.3%** rispetto al modello non ottimizzato;
2. **32 Cluster e centroidi a densità**: accuratezza, AUC e tempo di inferenza medio sono migliorati rispettivamente dello **0.6%**, **0.2%** e **7%** rispetto al modello di partenza.

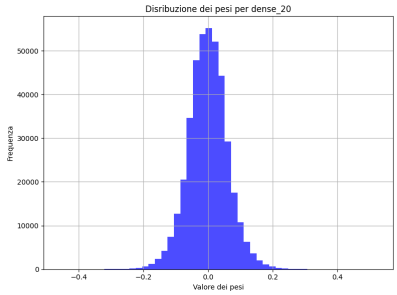
E' importante notare che alcuni modelli clusterizzati presentano valori di AUC e accuratezza leggermente superiori rispetto al modello non ottimizzato. Tale miglioramento può essere attribuito a una scelta ottimale del numero di cluster e delle modalità di inizializzazione dei centroidi, in grado di ridurre il rumore rispetto al modello di partenza. Inoltre, il fine-tuning su dati augmentati provenienti dal dataset AirtLab contribuisce ulteriormente a migliorare le prestazioni del modello. I due modelli clusterizzati sono comunque molto simili sia per tempo di inferenza, sia per AUC e sia per accuratezza: la differenza più importante è l'assorbimento energetico. La prima soluzione proposta, la clusterizzazione a centroidi lineari, è più efficiente rispetto alla seconda del **38.6%**.
E' a questo punto lecito chiedersi come mai tenendo fisso il numero di cluster e variando solo la tipologia di inizializzazione dei centroidi si hanno risultati mediamente migliori quando i centroidi sono inizializzati a densità. La risposta risiede nella distribuzione dei parametri nei vari strati, come riporta la figura 5, che non sono sparsi, ma per lo più accentrati attorno a uno specifico valore.



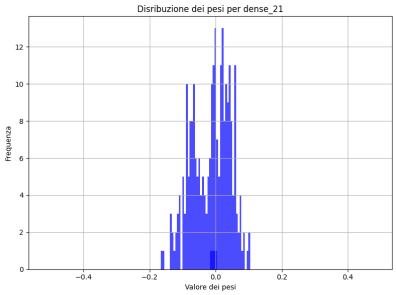
(a) Layer *Time Distributed con MobileNetV3 Small*.



(b) Layer *ConvLSTM*.



(c) Layer *Denso*.



(d) Layer *Denso*.

Fig. 5: Distribuzioni dei pesi della rete *MobileNetV3 Small + ConvLSTM*

Per quanto riguarda le tecniche di **pruning**, i risultati migliori sono stati registrati nel:

1. **Pruning al 50%**: rispetto al modello non ottimizzato, i valori di accuratezza e AUC sono diminuiti rispettivamente dello **0.5%** e **1.4%**, il tempo di inferenza medio è diminuito del **11.6%** e il consumo energetico medio è aumentato del **21.7%**;
2. **Pruning al 60%**: rispetto al modello non ottimizzato, i valori di accuratezza e AUC sono diminuiti rispettivamente del **6.1%** e **1%**, il tempo di inferenza medio è diminuito del **18.6%** e il consumo energetico medio è diminuito del **5.56%**, passando così da 1.8 Wh a 1.7 Wh.

L'accuratezza e l'AUC sono più alti e stabili nel modello con il pruning al 50% e, anche da un punto di vista energetico, questo modello è migliore del **26%** rispetto a quello con pruning al 60%. D'altra parte, il modello con il pruning al 60% è più veloce del **7.9%** rispetto al modello con sparsità finale del 50%. In questo caso, quindi, non esiste un modello migliore in assoluto, in quanto i due si bilanciano in termini di efficienza ed efficacia.
Infine, per quanto riguarda le tecniche miste di pruning e quantizzazione, si è verificato come il *pruning al 50% + QPT Float 16* sia migliore in termini di accuratezza (+**3.98%**), AUC (+**12.37**) e tempo di inferenza medio (-**14.3%**) e peggiore in termini di peso(+**45%**) e consumo energetico(+**21.7%**) rispetto al modello *pruning al 50% + QPT Float 16*.

4.2 MobileNetV3 Small + BiLSTM

Tecnica	Accuratezza (%)	AUC(%)	Peso modello (MB)	Tempo inferenza medio batch (s), *su 102 s di video in batch da 16 Frame*	Consumo energetico medio (Wh), *calcolato con 102 secondi di video*
Quantizzazione					
Intera a 8 bit con fallback Float	80	83	28.9	0.408	1.8
Intera a 8 bit	51	74.6	28.9	0.42	1.8
Float a 16 bit	90	97.7	57	0.278	1.7
Quantizzazione Aware Training	81.77	99.82	114.5	0.38	1.54
Clustering[5]					
8 Cluster e centroidi inizializzati a *densità*	80	81.3	558.4	0.47	4.5

[5] Il perchè del peso così elevato rispetto agli altri modelli, pur essendo una tecnica di ottimizzazione, è stato trattato al paragrafo 3.3

Tecnica	Accuratezza (%)	AUC(%)	Peso modello (MB)	Tempo inferenza medio batch (s), *su 102 s di video in batch da 16 Frame*	Consumo energetico medio (Wh), *calcolato con 102 secondi di video*
8 Cluster e centroidi inizializzati *lineari*	50.8	77	558.4	0.49	4.1
8 Cluster e centroidi inizializzati *random*	32	26	558.4	0.546	4.16
8 Cluster e centroidi inizializzati *Kmeans++*	50.8	76.87	558.4	0.48	3.7
16 Cluster e centroidi inizializzati a *densità*	53	81.7	558.4	0.75	5.6
Pruning					
Sparsità finale 40%	70.7	97.5	114.5	0.39	1.3
Sparsità finale 50%	74	97.6	114.5	0.37	2.0
Sparsità finale 60%	72.4	97.2	114.5	0.33	1.4
Sparsità finale 80%	73.5	97.2	114.5	0.36	1.5
Pruning + Quantizzazione					
Pruning 50% + QPT int con fallback Float	74.5	84.89	28.9	0.415	1.78
Pruning 50% + QPT Float 16	69.0	97.5	57.3	0.28	1.9

Table 7: Risultati ottenuti con MobileNetV3 Small + BiLSTM

In tabella 7 sono riportati i risultati ottenuti dall'inferenza sui vari modelli ottimizzati a partire dalla rete *MobileNetV3 + ConvLSTM*.

Per quanto riguarda la quantizzazione, come nel caso della sezione 4.1, le tecniche che producono i modelli con il compromesso che si ritiene migliore tra accuratezza e "leggerezza" del modello sono:

1. **Quantizzazione intera a 8 bit con fallback a Float**: è registrata una diminuizione di accuratezza e AUC rispettivamente del **10.6%** e del **14%** rispetto al modello originale non quantizzato, inoltre, anche il peso del modello, il tempo di inferenza medio e il consumo energetico medio sono diminuiti rispettivamente del **91.4%**, **41.7%** e **18.2%**: il punto di forza di questo modello è senza dubbio la sua elevata compressione unita con una relativamente bassa perdita di efficacia;
2. **Quantizzazione Float a 16 bit**: rispetto al modello non quantizzato, l'accuratezza è diminuita dello **0.6%**, mentre l'AUC è diminuita dello **0.14%**. D'altra parte, si registrano migliormenti in termini di peso, tempo di inferenza medio e consumo energetico medio, rispettivamente del **83%**, **60.3%** e **22.7%**.

Osservando i dati, si nota come il modello quantizzato a Float 16 bit risulti essere nettamente migliore rispetto al modello quantizzato a intero 8 bit con fallback a Float. L'unico problema risiede nel peso, dove viene misurato un aumento del **49.3%** nel passagio da Int a 8 bit con fallback a Float a Float a 16 bit. Se il dispositivo edge scelto avesse almeno 57/60 MB di memoria, a parità di quantizzazioni, la quantizzazione a Float a 16 bit sarebbe la migliore sotto ogni punto di vista.
Per quanto riguarda la clusterizzazione, ricordando ancora una volta che per quanto spiegato a sezione 3.3 il peso del modello non viene preso come un parametro indicatore di bontà del modello e di conseguenza nemmeno il consumo energetico medio, c'è una soluzione nettamente migliore delle altre:

1. **8 Cluster e centroidi a densità**: rispetto al modello di partenza si nota una diminuzione di accuratezza, AUC e tempo di inferenza medio rispettivamente del **11.69%**. **16.54%** e **32.86%**.

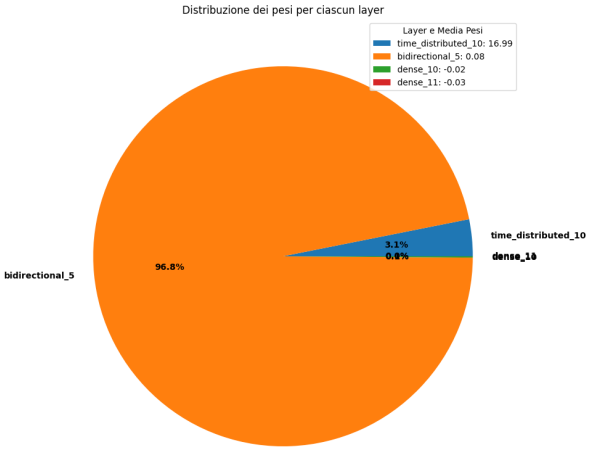
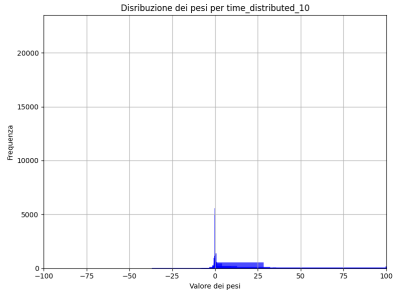
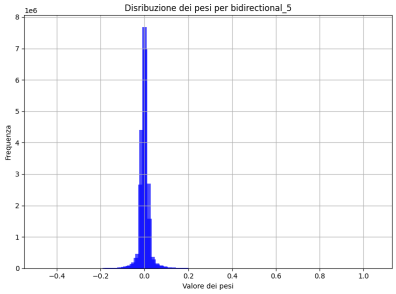


Fig. 6: Pesi nei vari layer del modello *MobileNetV3 Small + BiLSTM*.

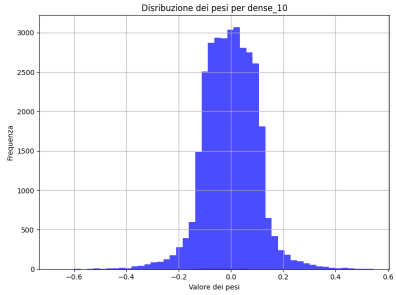
Come nel caso della clusterizzazione per la *MobileNetV3 Small + ConvLSTM* presentata al paragrafo 4.1, anche qui si nota un netto miglioramento quando, a parità di numero di cluster selezionati, viene scelto il metodo di inizializzazione dei centroidi a *densità*. La figura 7 evidenzia come le distribuzioni dei vari layer della rete siano "facilmente" scomponibili in cluster se si ragiona in termini di densità di pesi. Tale intuizione, unita con la scelta di un numero ragionevole di cluster, ha condotto, con tutti i limiti descritti al punto 3.3, a scegliere come metodo più efficace ed efficiente per la clusterizzazione quella con 8 cluster e inizializzazione dei centroidi a densità.



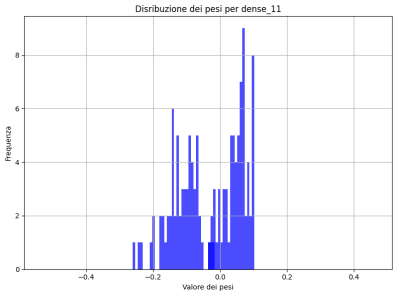
(a) Layer *Time Distributed*.



(b) Layer *Bidirectional*.



(c) Layer *Denso*



(d) Layer *Denso*

Fig. 7: Distribuzioni dei pesi della rete *MobileNetV3 Small + BiLSTM*

Per il *pruning*, invece, i modelli risultanti ottengono tutti valori simili, ma quelli che si credono migliori sono:

1. **Pruning al 50%**: rispetto al modello non ottimizzato, l'accuratezza e l'AUC sono diminuiti rispettivamente del **16.6%** e **0.24%**. Si sono registrati miglioramenti anche in termini di peso occupato su disco (**-66%**), tempo medio di inferenza (**-47%**) e consumo energetico medio (**-9%**);
2. **Pruning all'80%**: i valori di accuratezza e AUC sono diminuiti del **17%** e **0.64%** rispetto al modello di partenza, mentre si registra un miglioramento in termini di peso (**-66%**), tempo di inferenza medio (**-48.6%**) e consumo energetico medio (**-31.8%**).

La variabilità nell'accuratezza potrebbe essere causata da problemi di rumore. Come spiegato nella sezione 3.3, la potatura non elimina i pesi, ma li azzera. Dalla figura 6, si può osservare come la maggior parte dei pesi della rete iniziale siano presenti nella BiLSTM. Con la prunatura al 50% è probabile che siano stati azzerati i pesi che introducevano rumore nella rete, aumentando così in senso relativo l'efficacia, anche se rimane comunque molto inferiore rispetto a quella misurata con il modello di partenza.

Per quanto riguarda le tecniche miste, gli esperimenti hanno fornito i seguenti risultati:

1. **Pruning 50% + QPT Int con fallback FLoat**: L'accuratezza e l'AUC sono diminuiti rispettivamente del **16%** e **13%** rispetto al modello non ottimizzato. Inoltre, viene registrata una diminuzione del tempo di inferenza medio, consumo energetico medio e peso su disco rispettivamente di **40.7%**, **19%**.
2. **Pruning 50% + QPT Float16**:L'accuratezza e l'AUC sono diminuiti rispettivamente del **21.6%** e **0.34%** rispetto al modello non ottimizzato. Viene registrata inoltre una diminuzione del tempo di inferenza medio, consumo energetico medio e peso su disco rispettivamente di **60%**, **13.6%** e **83%**.

Occorre precisare che il calo di efficacia in termini di accuratezza rispetto al modello non ottimizzato è maggiore nel modello *Pruning 50% + QPT Float16* Tale fenomeno è da attribuirsi a errori di precisione numerica in fase di approssimazione tipici dei tipi di dato Float, che fanno così diminuire, seppur di poco, l'accuratezza.

5 Conclusioni e Sviluppi Futuri

Alla luce delle considerazioni fatte nelle sezioni 4.1 e 4.2, di seguito si procede alla definizione del modello migliore da trasferire su sistemi edge.
In generale, si nota che tutti i modelli, ottimizzati e non, che hanno nella loro architettura la BiLSTM sono peggiori rispetto a quelli con la ConvLSTM, non solo per applicazioni in edge, ma anche in generale: in linea di massima, infatti, hanno valori di accuratezza e AUC minori e valori di peso del modello, tempo di inferenza medio per batch e consumo energetico medio maggiori. Per questo motivo vengono esclusi dalla scelta.
In particolare, si ritiene che il modello migliore tra tutti quelli proposti sia quello dell'architettura MobileNetV3Small+ConvLSTM+FC con la tecnica di quantizzazione Float a 16 bit, che ottiene la combinazione di valori che sembrerebbe più conveniente: non solo mantiene un'accuratezza molto elevata nonostante la diminuzione del peso del 76.5%, ma riesce ad ottenere un tempo di inferenza breve e a diminuire il consumo energetico.

In conclusione del lavoro svolto si propongono idee per potenziali sviluppi futuri:

- Trasferimento del modello considerato migliore su sistemi edge (come la scheda Jetson Nano della Nvidia) tramite la conversione in formato ONNX per testare le sue effettive prestazioni;
- Cercare, tramite tecnica *NAS*[6](*Neural Architecture Search*), la combinazione di iperparametri che nelle tecniche di clustering e pruning produce risultati migliori;

[6]Neural Architecture Search (NAS) è una tecnica di deep learning che automatizza la progettazione delle architetture di rete neurale, esplorando un ampio spazio di possibili strutture attraverso strategie di ricerca come l'ottimizzazione bayesiana o l'apprendimento per rinforzo, con l'obiettivo di trovare configurazioni ottimali che migliorino le prestazioni dei modelli su specifici compiti.

- Risolvere il problema dell'esecuzione del metodo "`tfmot.clustering.keras.strip_clustering()`" riscontrato e descritto a sezione 3.3;
- Provare altre tipologie di backbone per l'architettura oltre la MobileNetV3 Small, come la MobileNetV4 [12] e la MobileOne [13], che richiedono l'implementazione non con la libreria Keras/TensorFlow come fatto in questo lavoro, ma con la libreria PyTorch.

References

[1] Avvenire: Rapine, violenze, risse: i reati compiuti dai minori sono in aumento (2020)

[2] Sernani, P., Falcionelli, N., Tomassini, S., Contardo, P., Dragoni, A.F.: Deep learning for automatic violence detection: Tests on the airtlab dataset. IEEE Access **9**, 160580–160595 (2021)

[3] AIRTLab: A-Dataset-for-Automatic-Violence-Detection-in-Videos. https://github.com/airtlab/A-Dataset-for-Automatic-Violence-Detection-in-Videos

[4] Hockey Fight Vidoes. https://www.kaggle.com/datasets/yassershrief/hockey-fight-vidoes

[5] T. Hassner, Y.I., Kliper-Gross, O.: Violent flows: Real-time detection of violent crowd behavior. In: 3rd IEEE International Workshop on Socially Intelligent Surveillance and Monitoring (SISM) at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2012). www.openu.ac.il/home/hassner/data/violentflows/

[6] Traoré, A., Akhloufi, M.A.: Violence detection in videos using deep recurrent and convolutional neural networks. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 154–159 (2020). https://doi.org/10.1109/SMC42975.2020.9282971

[7] Khan, M., Saddik, A.E., Gueaieb, W., De Masi, G., Karray, F.: Vd-net: An edge vision-based surveillance system for violence detection. IEEE Access **12**, 43796–43808 (2024) https://doi.org/10.1109/ACCESS.2024.3380192

[8] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520. IEEE Computer Society, Los Alamitos, CA, USA (2018). https://doi.org/10.1109/CVPR.2018.00474 . https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474

[9] Mohamed Elesawy, M.A.E.-M. Mohamed Hussein: Real Life Violence Situations Dataset. https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset

[10] Wu, P., Liu, J., Shi, Y., Sun, Y., Shao, F., Wu, Z., Yang, Z.: Not Only Look, but Also Listen: Learning Multimodal Violence Detection Under Weak Supervision. https://roc-ng.github.io/XD-Violence/

[11] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)

[12] Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F., Wang, W., Banbury, C., Ye, C., Akin, B., Aggarwal, V., Zhu, T., Moro, D., Howard, A.: MobileNetV4 – Universal Models for the Mobile Ecosystem (2024)

[13] Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: Mobileone: An improved one millisecond mobile backbone. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7907–7917 (2023)