Hurricane Damage Detection

Matteo Denova, 813101 · Simone Ferro, 813660 · Davide Zulato, 876101

Febbraio 28, 2022

Abstract

Con la larga disponibilità di immagini satellitari assume sempre maggior rilevanza un'analisi efficace di numerose immagini con il supporto di tecniche di deep learning. Questa disciplina ha visto negli ultimi anni un importante sviluppo nel campo dell'intelligenza artificiale e viene sempre più impiegata per tasks di image classification. All'interno di questo lavoro l'utilizzo di reti deep sarà un elemento chiave nel supporto alle decisioni dopo un'emergenza, in particolare a seguito dell'Uragano Harvey che ha colpito il Texas nel 2017. In questo paper ci siamo occupati di costruire e allenare una rete neurale in grado di identificare in modo autonomo se una data area presenti o meno danni dovuti all'allagamento. La rete allenata è in grado di identificare se un'area sia stata o meno danneggiata con un'accuracy prossima al 98%. il grande vantaggio di questa tecnica è che non ha richiesto specifiche *domain knowledges*.

1 Introduzione

All'interno di questo lavoro presentiamo l'architettura di una deep convolutional neural network per identificare (i.e. classificazione binaria) i danni provocati da un uragano senza doverci affidare ad una *domain knowledge* terza necessaria durante l'estrazione delle features o ricorrere a dispendiosi processi di feature engineering. Le features rilevanti per la stima dei danni alle strutture sono infatti estratte automaticamente da una serie di feature maps ad ogni passo dello sviluppo della rete.

Nel campo della classificazione delle immagini le reti neurali e le CNNs in particolare trovano ampio utilizzo e i campi applicativi spaziano dalla classificazione di malattie, a partire da risonanze magnetiche in ambito medico, a utilizzi in ambito agricolo.

Questo paper è organizzato come segue: il paragrafo successivo, Materiali e Metodi, fornisce informazioni riguardo alle CNNs profonde e ne giustifica la scelta per questo task. La sezione Materiali in particolare fornisce indicazioni sul dataset in oggetto mentre la sezione metodi spiega anche l'architettura della nostra rete e i layers utilizzati. I risultati,

le rappresentazioni grafiche e le performances sono mostrate al lettore nella sezione 3) Risultati. La discussione dei problemi riscontrati e le limitazioni del nostro modello sono coperte dalla sezione 4) che conclude il lavoro.

2 Materiali e metodi

1. Materiali

Il dataset in esame è composto da immagini satellitari successive all'Uragano che ha colpito il Texas nel 2017. Le immagini sono divise in 2 gruppi (damage and no-damage). L'obiettivo è di implementare un modello che sia in grado di identificare automaticamente se una regione ha subito danni legati all'allagamento.

Area danneggiata

Area non danneggiata

Il **dataset** oggetto dell'analisi proviene dalla piattaforma Kaggle e riporta immagini satellitari successive all'uragano Harvey che ha colpito la zona di Houston nel 2017. In questo paper la rete viene allenata su un ampio campione di training; l'errore di generalizzazione viene calcolato attraverso un validation set e infine la performance della rete viene valutata sul test set. All'interno del dataset è presente sia un test set bilanciato che un test set sbilanciato. Il test sbilanciato è composto da 1000 non danneggiati e da 8000 danneggiati. Il test bilanciato è invece composto da 2000 immagini equamente divise tra danneggiate e non danneggiate.

2. Metodi

In questa sottosezione presentiamo gli approcci utilizzati.

La scelta del metodo da utilizzare per la classificazione è legata alla natura del problema: abbiamo a che fare con immagini satellitari di edifici e terreni, i quali hanno diverse rotazioni e traslazioni spaziali: per questa ragione è necessario fare ricorso a un modello che sia in grado di cogliere i patterns dell'immagine in modo invariante rispetto a rotazioni e traslazioni spaziali. Perciò abbiamo ritenuto opportuno utilizzare Convolutional Neural Networks.

L'architettura della rete segue la classica struttura delle reti convoluzionali: alterniamo convolutional layers a pooling layers (Max pooling). Gli ultimi strati seguono una struttura FFNN (Feed Forward Neural Network) e sono quelli che operano la classificazione binaria; l'ultimo strato, poiché ci confrontiamo con un task di classificazione binaria, è formato da una funzione di attivazione di tipo sigmoidale.

La differenza fondamentale tra "densely connected layer" e "convolution layer" è che il primo acquisisce caratteristiche globali nell'input feature space, mentre il secondo impara (*learn*) patterns locali: nel caso delle immagini i patterns specifici vengono acquisiti attraverso finestre 2D.

La scelta di utilizzare CNN per l'analisi delle immagini satellitari dei danni conseguenti all' uragano è legata a due importanti proprietà:

I patterns acquisiti sono invarianti a traslazioni.
I convolutional layers possono apprendere gerarchie spaziali dei patterns.

La proprietà dell'invarianza rispetto a traslazioni dei patterns acquisiti dalla rete è estremamente importante ai fini di questo paper poiché una volta imparata una certa caratteristica, ad esempio nell'angolo dell'immagine in input, la rete è in grado di riconoscerla ovunque. Avendo a che fare con immagini satellitari e con danni ad edifici collocati in punti diversi dell'immagine, questa proprietà delle reti convoluzionali offre un notevole vantaggio rispetto ad una rete densely connected, la quale dovrebbe al contrario ri-imparare un pattern se posizionato in un punto diverso dell'immagine. (Chollet p.115) Per queste ragioni abbiamo ritenuto opportuno utilizzare *convnets* per questo paper.

Preprocessing

Le immagini satellitari all'interno del dataset sono in formato JPEG; prima di procedere con l'architettura e l'allenamento della rete ci siamo occupati di formattare le immagini in modo da essere tensori 3D analizzabili dalla rete attraverso i seguenti passaggi:

- Lettura delle immagini dal dataset
- Ricodifica da .jpeg a griglia di pixels RGB
- Conversione di queste ultime in tensori 3D
- Riscalare i pixel e portare le immagini allo stesso intervallo

Una delle principali questioni con cui ci siamo confrontati in questo task di classificazione riguarda il rapporto tra ottimizzazione e generalizzazione. Nel Machine-Learning, con ottimizzazione si fa riferimento al processo di aggiustamento del modello per ottenere le migliori prestazioni possibili nel training, mentre la generalizzazione concerne la capacità del modello allenato di performare bene su dati che non ha ancora visto.

All'interno di questo paper l'analisi è svolta su un vasto campione di immagini satellitari; per questa ragione sembrerebbe non necessario utilizzare tecniche di data augmentation, utili al fine di mitigare il fenomeno di overfitting legato all'utilizzo di piccoli training samples. La data augmentation applica trasformazioni casuali alle immagini del training set al fine di aumentare la numerosità campionaria e allenare la rete su un numero maggiore di immagini. Come mostreremo successivamente nella tabella 1, l'utilizzo di questa tecnica non ha portato a sostanziali miglioramenti delle performance della rete, coerentemente con le aspettative iniziali.

Approcci utilizzati per l'image classification.

Come vedremo anche nei risultati, per affrontare il problema di classificazione di aree danneggiate dall'uragano Harvey abbiamo in primo luogo allenato una rete da zero (from the *scratch*) per poi provare ad avvalerci di reti pre-allenate con l'obiettivo di migliorare le performance. Tuttavia, esse non hanno portato a miglioramenti in termini di accuracy. La letteratura accademica che negli ultimi anni si è occupata di image classification e, più in generale, di computer vision si è spesso avvalsa di diverse reti pre-allenate, disponibili sia nei pacchetti (e.g VGG16, VGG19, MobileNet, ResNet50, Inception V3, Xception all'interno di Keras) che messe a disposizione da altri ricercatori dopo averle allenate su ampi datasets.

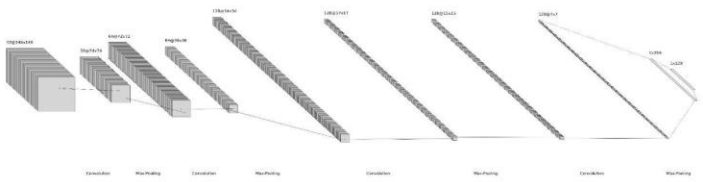
L'immagine che segue mostra l'architettura della rete convoluzionale utilizzata.

Figura 1

```
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## conv2d_3 (Conv2D)            (None, 148, 148, 32)      896
## _________________________________________________________________
## max_pooling2d_3 (MaxPooling2D)  (None, 74, 74, 32)     0
## _________________________________________________________________
## conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
## _________________________________________________________________
## max_pooling2d_2 (MaxPooling2D)  (None, 36, 36, 64)     0
## _________________________________________________________________
## conv2d_1 (Conv2D)            (None, 34, 34, 128)       73856
## _________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)  (None, 17, 17, 128)    0
## _________________________________________________________________
## conv2d (Conv2D)              (None, 15, 15, 128)       147584
## _________________________________________________________________
## max_pooling2d (MaxPooling2D)  (None, 7, 7, 128)        0
## _________________________________________________________________
## flatten (Flatten)            (None, 6272)              0
## _________________________________________________________________
## dropout (Dropout)            (None, 6272)              0
## _________________________________________________________________
## dense_1 (Dense)              (None, 512)               3211776
## _________________________________________________________________
## dense (Dense)                (None, 1)                 513
## =================================================================
## Total params: 3,453,121
## Trainable params: 3,453,121
## Non-trainable params: 0
```

L’architettura della CNN utilizzata presenta quattro layers convoluzionali alternati ad altrettanti layers di pooling, seguono un layer di flatten, a cui viene aggiunto un layer di dropout e due layers densi.

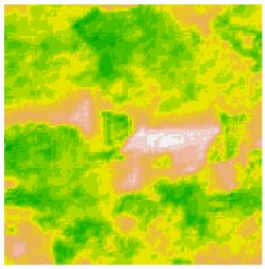
Figura 2



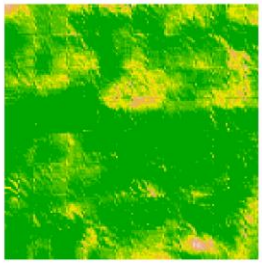
L’immagine precedente (Fig.2) mostra l’architettura della rete convoluzionale utilizzata come mostrata dall’output di R in fig.1

Vediamo ora come l’architettura della nostra rete trasforma l’immagine per catturarne i patterns che permettono la classificazione binaria: per farlo, consideriamo la seguente immagine non allenata dal modello, presa dal test e appartenente al gruppo “Damage” che rappresenta quindi un’area danneggiata data in input alla rete.

Visualizziamo poi le feature maps con tre dimensioni: larghezza, altezza e profondità (channels). Ogni canale codifica features indipendenti, perciò, siamo in grado di visualizzare le feature maps attraverso immagini in 2D per ogni channel (i.e. *depth* axis).



Secondo canale

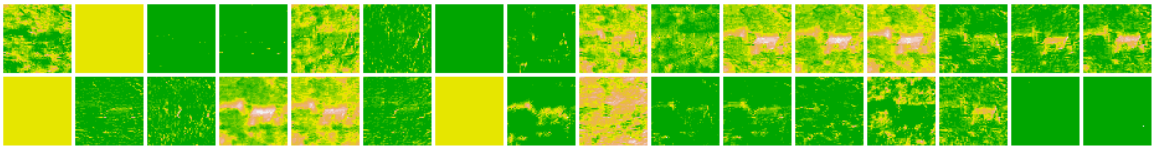


Settimo canale

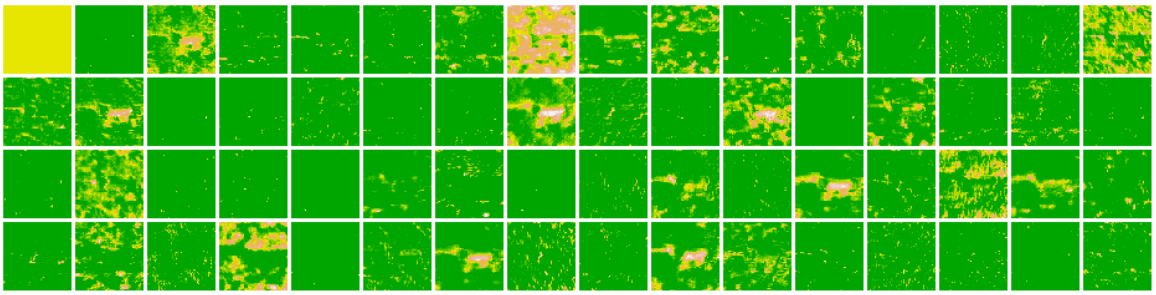
Le due immagini rappresentano 2 dei 32 canali appartenenti al primo layer (come osserviamo nell'architettura è una feature map 148x148 con 32 canali), osserviamo che i canali presi ad esempio codificano aspetti diversi dell'input.

Le immagini che seguono rappresentano invece parte della visualizzazione completa di tutte le attivazioni della rete, sono state estratte e plottate tutti i canali in ognuna delle activation maps prese in considerazione. Sono state prese a scopo dimostrativo 3 outputs intermedi da layers convoluzionali con 32, 64 e 128 channels. Ricordiamo che con l'aumentare della profondità diminuiscono le altre 2 dimensioni dell'output.

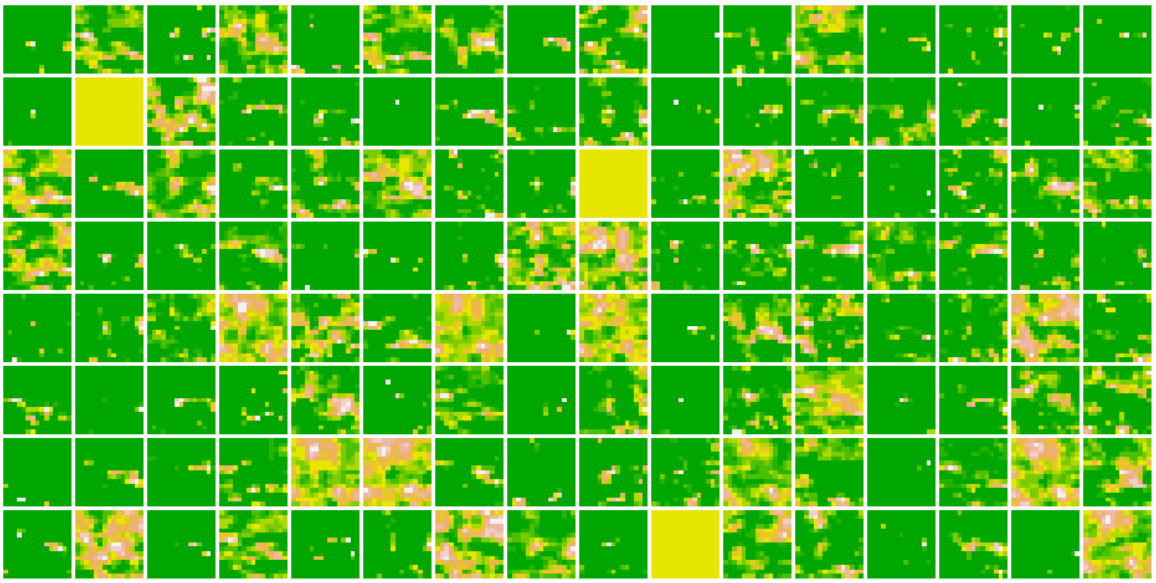
Damage activations 2 conv 2d_15 (32 canali)



Damage activations 3 conv 2d_14 (64 canali)



Damage activations 7 conv 2d_12 (128 canali)



Notiamo che negli strati più profondi della rete le attivazioni diventano sempre più astratte e difficili da interpretare per l'occhio umano. Questo perché mentre i primi strati contengono ancora molte informazioni sull'input e catturano aspetti più generici, a livelli più alti sono contenute sempre meno informazioni dell'immagine iniziale e sempre più informazioni che riguardano la classe dell'immagine.

3 Risultati

In questa sezione saranno presentati i risultati ottenuti durante l'analisi. La tabella seguente (tabella 1) sintetizza i risultati ottenuti con diverse tecniche di regolarizzazione (dropout in particolare), Data Augmetation e la sperimentazione di diversi algoritmi di ottimizzazione. Il criterio di scelta del modello è sulla base dell'accuracy sia sul test set bilanciato che sul test set sbilanciato. La tabella seguente riporta anche loss e accuracy nel validation. Abbiamo allenato le nostre reti utilizzando la libreria Keras con TensorFlow in backend, la dimensione del mini batch per la discesa del gradiente è 32.

Tabella 1

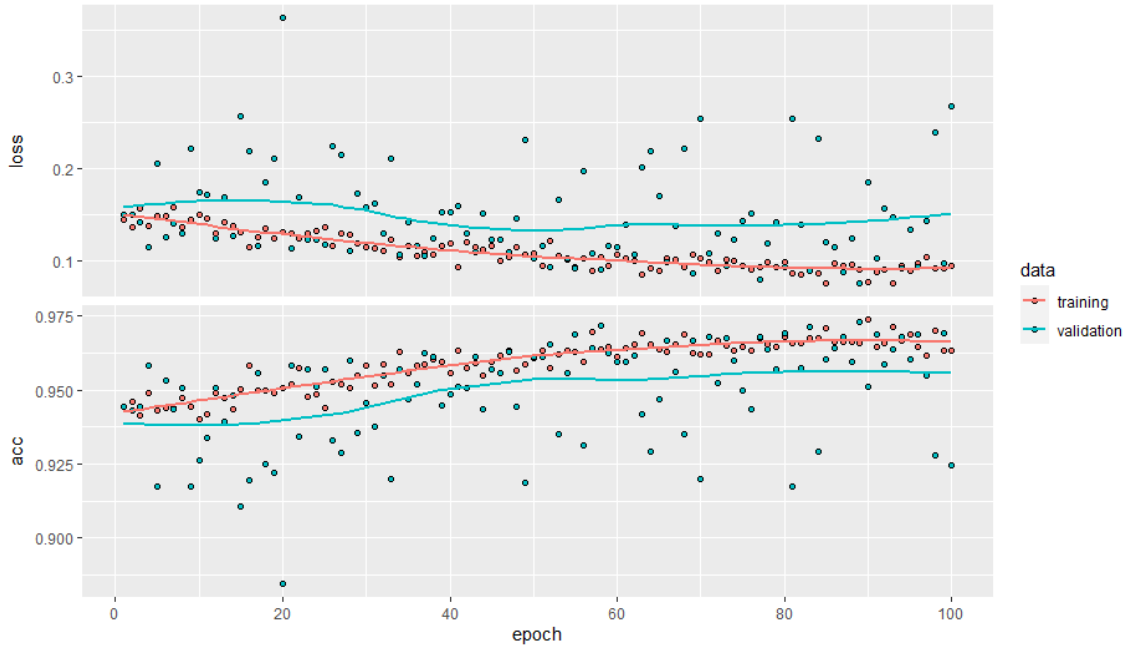
Model	Validation Accuracy (Loss)	Test Accuracy (bilanciato)	Test Accuracy (non bilanciato)
CNN	0.95 (0.124)	0.949	0.956
CNN+DA	0.924 (0.268)	0.923	0.985
CNN+DO (0.6)	0.9494 (0.128)	0.954	0.957
CNN+DO (0.5)	**0.972 (0.078)**	**0.977**	**0.983**
CNN+DA+ DO(0.5)	0.953 (0.175)	0.958	0.967
CNN+DA+DO (ADAM)	0.952 (0.146)	0.956	0.985

Dopo aver allenato e testato i modelli, quelli con le performance migliori sono risultati essere CNN+DA, CNN+DO (0.5), CNN+DA+DO (ADAM). Per scegliere il migliore, li abbiamo valutati singolarmente e confrontati su diversi aspetti.

CNN con data augmentation (CNN+DA)

La figura seguente mostra l'andamento delle curve di loss e accuracy nel training set e nel validation set per quanto riguarda questa rete.

Figura 3

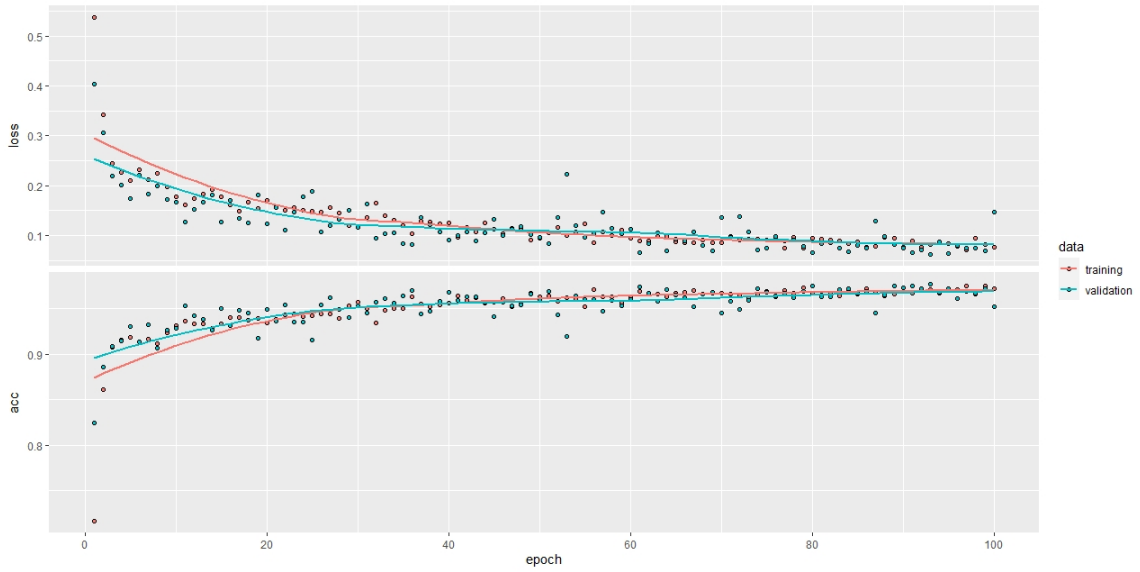


Come si osserva dalla figura fig.3 l'andamento della accuracy e della loss in training e validation presenta un path instabile e mostra un leggero overfitting, poiché nell'accuracy la curva del training rimane stabilmente sopra a quella del validation, abbiamo quindi deciso di escluderlo dalle possibili scelte.

CNN+DA+DO con Adam Optimizer

Escluso il modello precedente, il modello migliore è da ricercare fra CNN+DA+DO con Adam e CNN+ DO.
La seguente immagine mostra loss e accuracy in training e validation set per quanto riguarda la rete CNN con dropout (50%), data augmentation e algoritmo di ottimizzazione "Adam". L' optimizer è quello che determina come si aggiorna la rete sulla base della loss function, l'algoritmo di ottimizzazione Adam ("adaptive moment estimation") è da vedersi come un'estensione della discesa del gradiente: mentre nel classico algoritmo di discesa del gradiente il learning rate è costante e specificato all'interno del modello, grazie a questo ottimizzatore non è necessario specificare il learning rate.

Figura 4



Dopo aver allenato la rete per 100 epoche e con 100 steps per epoca otteniamo una training loss pari a 0.0764, un'accuracy di 0.9725, una validation loss di 0.1465 e una validation accuracy pari a 0.9519.

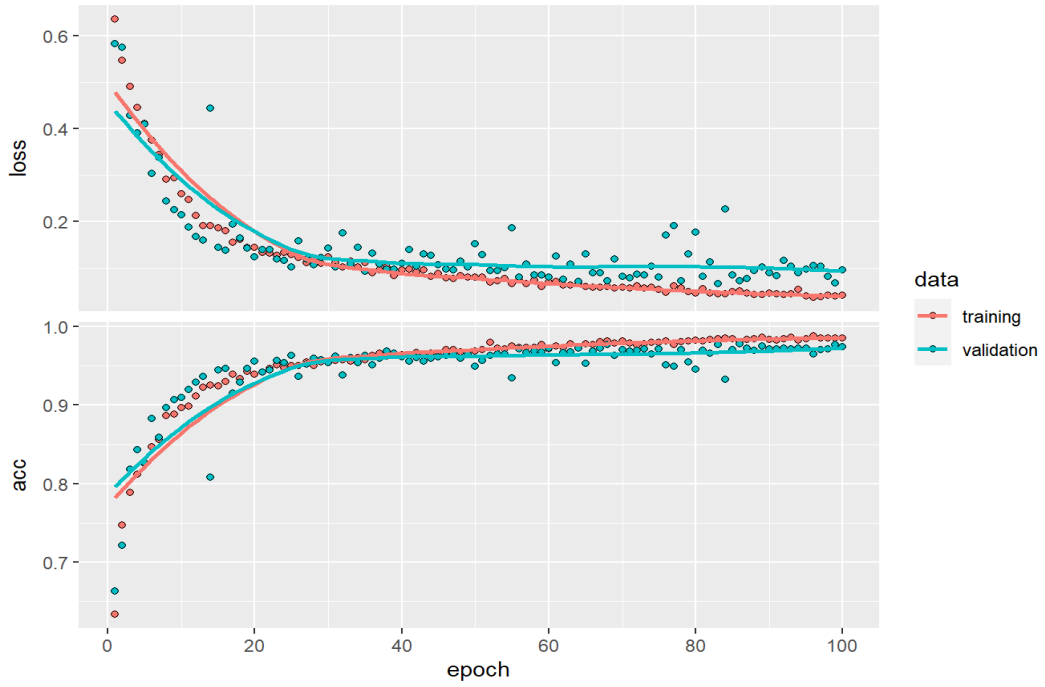
Rispetto alla CNN con dropout (0.5), l'utilizzo dell'Adam optimizer ha portato ad un leggero miglioramento delle performances per quanto riguarda il test set non bilanciato (accuracy 0.985) ma non sul test set bilanciato in cui i risultati peggiorano: in un'applicazione reale non è nota l proporzione di aree danneggiate e non, quindi è preferibile un modello che abbia performance più omogenee possibili al variare della proporzione delle classi.

Per questa ragione abbiamo scelto come miglior modello la rete CNN con dropout al 50% e optimizer RMSprop, il learning rate specificato è 10^{-4} .

CNN+dropout (0,5)

Il modello scelto è quindi CNN+dropout (0.5) e ottimizzatore RMSprop, che presenta un'accuracy pari a 0.977 sul test set bilanciato e 0.983 sul test set non bilanciato.

Figura 5



In conclusione, il miglior modello trovato è la Convolutional Neural Network con dropout del 50% visti i risultati più stabili nei test bilanciato e sbilanciato.

4 Discussioni

In questa sezione ci occupiamo della discussione dei problemi riscontrati e delle limitazioni del nostro modello per poi passare alle possibili direzioni future di questo lavoro sia dal punto di vista metodologico sia per quanto concerne altri campi di applicazione.

In letteratura, l'utilizzo di reti pre-allenate permette non solo di avere performances migliori su piccoli campioni (Chollet, p.132) ma anche di beneficiare del lavoro di altri ricercatori e attraverso l'utilizzo di tecniche come il fine tuning, apportare miglioramenti con il conseguente progresso dell'ambito applicativo. In questo contesto l'utilizzo di reti pre-allenate non ha portato ad un miglioramento delle performances e in particolare dell'accuracy del nostro modello. Uno dei limiti di questo lavoro riguarda infatti la mancanza di miglioramenti delle performances sulla base di reti pre-allenate. Il miglior modello trovato è una CNN con drop-out 0.5 come riportato nella tabella 1. Le migliori performances sono quindi il risultato dell'applicazione di una rete convoluzionale allenata da zero con un layer di dropout che svolge il ruolo di regolarizzatore ed evita che si verifichi un overfitting eccessivo.

L'utilizzo di queste tecniche per lo studio dei danni e la conseguente allocazione di risorse durante i disastri naturali è da vedersi come un proficuo campo di applicazione per questa disciplina. Nel contesto italiano, dove al contrario degli stati uniti è presente l'organo della protezione civile, la presenza di queste risorse è un'opportunità che permetterebbe di allocare risorse in modo efficace durante un periodo di emergenza, evitando sopralluoghi con mezzi di terra a favore di droni ed immagini satellitari, in sinergia con i metodi e le tecniche di deep learning. Le reti pre-allenate e le reti deep ci permettono in un certo senso di "*salire sulle spalle dei giganti*" (Per usare una famosa metafora di Bernardo di Chartres) e di raggiungere, grazie a conoscenze pregresse, una migliore comprensione dei fenomeni. Il nostro contributo in particolare è stato possibile grazie alle tecniche di deep learning che permettono questa comprensione anche senza una solida domain knowledge del fenomeno. Questo aspetto non svaluta il lavoro degli esperti ma piuttosto ne democratizza le conoscenze rendendole utili, interdisciplinari e disponibili ad un pubblico più vasto.

Sviluppi futuri di questo lavoro dal punto di vista applicativo potrebbero essere legati allo studio post-cataclisma di di disastri naturali nel nostro paese; nonostante il nostro territorio non sia soggetto a danni provocati da uragani, terremoti e nubifragi colpiscono periodicamente le nostre regioni e un utilizzo delle tecniche presentate potrebbe porsi come un utile strumento a favore della collettività.

5 Bibliografia

Francois Chollet and J.J. Allaire, “Deep Learning With R”, Manning 2018

Quoc Dung Cao and Youngjun Choe, “Building Damage Annotation on Post-Hurricane Satellite Imagery Based on Convolutional Neural Networks”. *Nat Hazards* 103**,** 3357–3376 (2020). https://doi.org/10.1007/s11069-020-04133-2

Karen Simonyan and Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, https://arxiv.org/abs/1409.1556

Keras: Keras https://keras.io/