SHALLOW WATER EMULATION WITH NEURAL NETWORKS FOR DISASTER PREVENTION

Diego Calanzone
Università degli Studi di Parma
diego.calanzone@studenti.unipr.it

ABSTRACT

Le simulazioni Shallow Water consentono lo studio delle dinamiche dei fluidi in disastri naturali o contesti controllati. Nonostante la presenza di accurati modelli fisici di simulazione (es: *LISFLOOD-FP*), il costo computazionale rende il loro utilizzo oneroso e non adatto per previsioni real-time. Con la crescente importanza degli algoritmi di apprendimento automatico, come le reti neurali, studiamo in questo articolo la natura dei dati di simulazione, proponendo un'architettura in grado di ricostruire le dinamiche in caso di breccia su una superficie. La simulazione di questa classe di eventi trova rappresentazione in due principali forme:

- **Smoothed Particle Hydrodynamics (SPH)**: rappresentazione di reticolati variabili di particelle su 3 dimensioni, con forze di reazione tra di esse.
- **Shallow Water Emulations (SWE)**: rappresentazione di una superficie attraverso griglie bi-dimensionali di risoluzione variabile. Ogni cella rappresenta un'area di fluido quantificata in termini di: velocitá del fluido X, Y; profonditá della colonna di fluido; elevazione del terreno di fondo.

Le *Shallow Water Emulations* saranno il soggetto di questo studio. L'obbiettivo principale é quello di simulare correttamente l'evoluzione di un'inondazione nel futuro in 3 modalitá:

1. Previsione di una sequenza di **N** frames nel futuro
2. Classificazione delle zone di superficie a rischio/non a rischio al tempo Δt
3. Previsione del livello di allagamento al tempo Δt

1. STATO DELL'ARTE

Questa classe di problemi di fluidodinamica é stata soggetto di studi in varie aree: dalla computazione parallela al deep learning.

- **Modello dei dati:** abbiamo utilizzato la rappresentazione proposta dal *Dr. Alessandro Dal Palú (2014) [1]*: una griglia bi-dimensionale con risoluzione variabile, predisposta per la scomposizione di dominio per blocchi e la computazione parallela GPU-Based con CUDA.

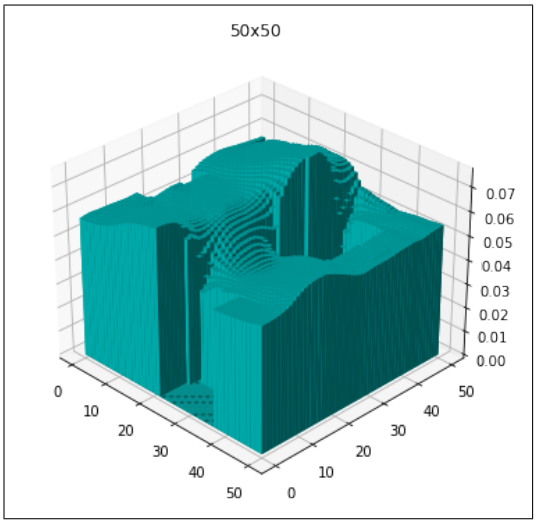

Figure 1. Frame di simulazione SWE, visualizzazione 3d della griglia di colonne di fluido.

- **Introduzione alle reti neurali:** l'utilizzo delle misure di portata della Breccia variabile é state introdotto da *M. Varesi (2018) [2]*, con l'applicazione di una rete feed forward semplice per lo studio del comportamento del fiume Secchia in Emilia Romagna.
- **Reti Neurali Convoluzionali:** con l'utilizzo del simulatore *LISFLOOD-FP*, *S. Kabir, S. Patidar, X. Xia e Q. Liang (2020) [3]* introducono un modello predittivo in grado di processare dati di simulazione sotto forma di fotogrammi. Oltre all'utilizzo di Support Vector Regression, vengono implementate le Convolutional Neural Networks, ideate proprio per il processing di immagini e in grado di identificare pattern tra porzioni di pixel prossimi tra loro.
- **Il simulatore PARFLOOD:** *F. Vignali [4]* introduce il simulatore *PARFLOOD*, sviluppato all'Universitá di Parma, analizzandone le prestazioni e l'accuratezza nel contesto del calcolo parallelo.
- **Next frame prediction:** nel paper di *C. Vondrick e A.Torralba (2016) [5]* si affronta il problema della previsione di frame nel futuro (soggetti in movimento, azioni, trasformazioni) attraverso la stima di differenze tra pixel dei frame temporali. Viene introdotto l'utilizzo dei "trasformatori" (approccio in

futuro rinominato anche *Residual Network*).

- **I Vision Transformers:** il team Google Brain Research propone una nuova architettura generica applicata alle immagini *A. Dosovitskiy, N. Houlsby (2020) [6]*. Eliminando il bias induttivo, questo modello sostituisce completamente le reti Convoluzionali, introducento un meccanismo di focus sulle feature importanti appreso durante il training, chiamato *Attention [7]*. I Transformers trovano oggi largo utilizzo nel Natural Language Processing, sorpassando in performance le architetture alternative.

2. STRUMENTI UTILIZZATI

- **Hardware** :

Type	Hardware	Amount
Personal	NVIDIA GTX 1070	1
Cloud	NVIDIA Tesla K80	1
Academic	NVIDIA Tesla P100	7
Cloud	Google TPU TF 2.1	1

- **Ambiente di sviluppo**: Python 3.5, Anaconda 4.5.11, Jupyter 6.1.5, WeightsBiases (wandb.ai) model optimization
- **Librerie di Machine Learning**: Tensorflow 2, PyTorch 1.71, Statsmodels 0.12.1, Numpy 1.19.5, Matplotlib 3.3.3
- **Documentazione**: Weights&Biases [12], Pierian Data Science [11], PyTorch 1.7.1 [10], Keras.io [9], Tensorflow.org [8]

3. GENERAZIONE DEI DATI

Per l'addestramento, abbiamo individuato come sorgente dati il software di simulazione PARFLOOD, sviluppato all'interno del *Dipartimento di Ingegneria e Architettura* dell'Universitá degli Studi di Parma [14].

Il simulatore **S** riceve in input piú mappe, rappresentanti il suolo di una reale area geografica **A**. In questo studio si analizza un'area nel percorso del fiume Toce, in Piemonte.
Si descrivono i file e parametri in input:

- File .BTM: batimetria, topologia del fondale
- File .MAN: scabrezza (rapporto rugositá/diametro della condotta) espressa secondo il coefficiente di Manning
- File .INH: condizioni iniziali
- File .BLN: condizioni al contorno (bordi, superfici, solidi)
- File .BCC: dettagli aggiuntivi legati al file .BLN
- File configurazione: durata simulazione (ore), granularitá campionamento (ore), parametri per le equazioni di simulazione.

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \nabla \cdot T + f$$

Figure 2. Equazione principale di Navier-Stokes

Stabilendo la posizione e l'intensitá di una breccia **B** (sorgente di flusso), **S** calcola le equazioni di Navier-Stokes [13], generando una sequenza di *N* frames temporali:

$$N = \frac{t_{simulazione}}{Granularity} \quad (1)$$

Dove la granularitá consiste nella frequenza di campionamento dei frame di simulazione, fissata a **0.2s**.
Ogni frame descrive l'area geografica con 3 matrici di misurazioni: **.DEP**, profonditá delle colonne d'acqua per cella; **.VVX**, velocitá del flusso nelle ascisse per cella; **.VVY**, velocitá del flusso nelle ordinate per cella. L'insieme di frames **N** formano una simulazione **K**, accompagnata da un file riassuntivo: **.MAXWSE**, matrice di massime altezze delle colonne d'acqua (per ogni cella).

4. CALCOLO DISTRIBUITO

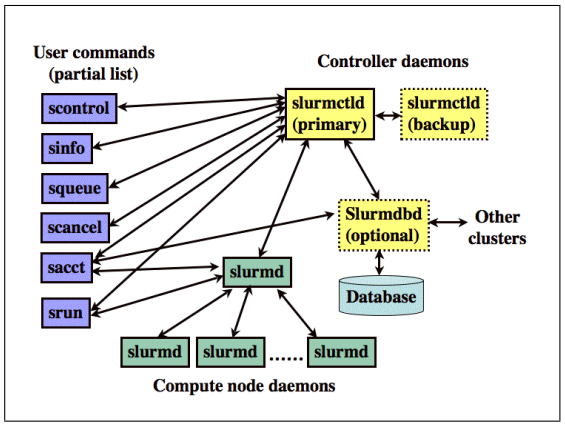

Figure 3. Architettura Slurm Workload Manager

Per ottenere un dataset eterogeneo e sufficientemente ricco, abbiamo eseguito **100 simulazioni** sulla stessa area geografica, introducendo **100 brecce** con intensitá differente.
Riassumiamo il framework in 6 passaggi:

1. Accesso al cluster HPC dell'Universitá di Parma
2. Generazione di 100 file BCC (portata) con differenti intensitá di breccia
3. Scheduling delle 100 simulazioni attraverso job paralleli gestiti dal Workload Manager *Slurm*
4. Avvio della decodifica dei file generati attraverso la submission di un secondo job
5. Preprocessing dei file e generazione di tensor n-dimensionali serializzati in formato *.numpy*.

5. DATA UNDERSTANDING

Attraverso la conoscenza di dominio, sono state identificate le seguenti variabili e i corrispettivi livelli dei dati:

- Portata breccia *B* **(feature)**: continua discreta, livello dei rapporti
- Elevazione terreno *BTM* **(feature)**: continua discreta, livello dei rapporti
- Altezza colonna d'acqua *DEP* **(feature, target)**: continua discreta, livello dei rapporti
- Allagamento oltre soglia *BIN* **(target)**: categorica dicotomica

Applicando un'analisi univariata, abbiamo analizzato i valori per **B** e **DEP**, considerando **BTM** costante, in quanto set di misurazioni in input non generate o calcolate dal simulatore.

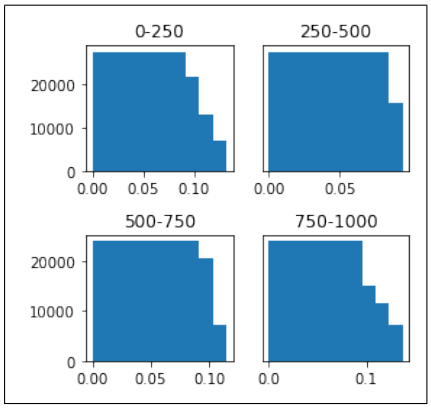

Figure 4. Distribuzione DEP su 4 intervalli temporali.

I valori DEP sono distribuiti tra **0.00** e **0.14**, in cui il centro dei dati (media) varia nel tempo, rispecchiando un flusso crescente e poi decrescente (**Figura 2**).

La portata della breccia B é una sequenza temporale di misurazioni da inizio a fine simulazione. In ciascuna delle 100 simulazioni abbiamo fissato una portata di picco differente, ottenendo diversi scenari. La **Figura 3** illustra differenti idrogrammi (valore medio DEP) in base a tre portate di picco variabili.

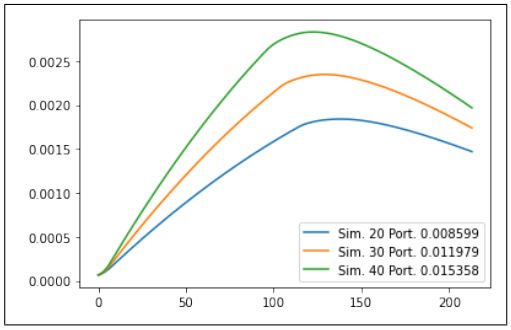

Figure 5. Progressione DEP media delle simulazioni 20, 30, 40.

Come illustrato, la portata della breccia **B** in ingresso influenza l'altezza delle colonne d'acqua **DEP** (target), giustificando la scelta delle features.

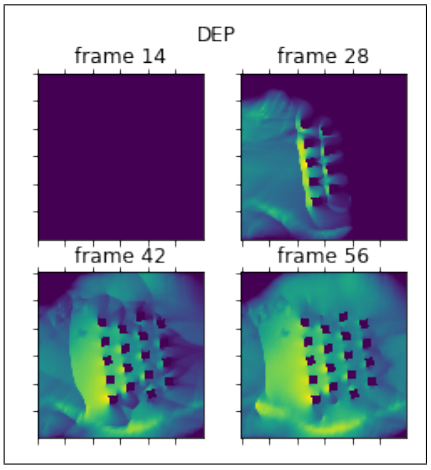

Figure 6. Visualizzazione di una matrice DEP nel tempo

Osservando la **Figura 4**, sorgono piú domande riguardo gli scenari su cui addestrare il modello predittivo:

- Fase iniziale di "inondazione" del bacino, in cui si passa da un terreno asciutto al pieno flusso. In questa fase il liquido colpisce gli ostacoli, quindi si accumula prima di distribuirsi nella superficie.
- Fase di distribuzione del fluido, in cui il flusso tende a stabilizzarsi (considerando una sorgente costante)
- Fase di eventuale deflusso, nel caso in cui la sorgente (punto di breccia) non sia costante, e quindi il livello dell'acqua tenderá ad abbassarsi progressivamente.

6. DATA PREPROCESSING

In base agli obiettivi precedentemente definiti (1, 2, 3), abbiamo sottoposto il dataset a diverse fasi di preprocessing:

- **Task (1, 2, 3)**: normalizzazione dei valori utilizzando la media μ e la deviazione std σ:

$$x = \frac{(x - \mu)}{\sigma} \tag{2}$$

- **Task (2)**: discretizzazione della matrice target MAXWSE per previsioni dicotomiche. Considerata una soglia di 1mm/5mm, ogni cella con valore superiore é considerata **true** (rischio di allagamento).
- **Task (1)**: interpolazione dei valori, sperimentata per rappresentare la fase iniziale di allagamento su piú frames, arricchendo il dataset. Si definiscono $\alpha = 0.5$ il learning rate, x_i il frame al tempo i, x_{i+1} il successivo, correlati dalla seguente formula:

$$x_i = (1 - \alpha) * x_{i-1} + \alpha * x_{i+1} \tag{3}$$

Tuttavia é presente un limite superiore: applicando oltre 2/3 iterazioni il seguente metodo, la differenza tra frames successivi diminuisce sempre di piú, peggiorando il potere predittivo del modello.

- **Task (1)**: cropping del frame, eliminando la superificie esterna al bacino e selezionando una sezione centrale 300x300, dove son presenti ostacoli.
- **Task (1, 3)**: eliminazione dei valori FLAG usati dal simulatore per indicare l'assenza di fluido (es: $x_{DEP} >= 1.E + 5$) :

- **Task (1)**: ricerca della stazionaritá attraverso differenziazione, richiesta per l'applicazione del modello ARIMA.
- **Task (1, 2, 3)**: features selection eseguita mediante layer convoluzionali. La rete neurale convoluzionale applica, per ogni layer, diversi filtri (matrici kernel con learnable weights) per l'estrazione di features quali bordi orizzontali, verticali, movimenti.

Per la costruzione del dataset abbiamo adottato principalmente la **Holdout Strategy**, verificando differenti combinazioni di parametri attraverso la **Grid Search**.

7. MODELING

- **Task 1 - ARIMA** (Auto Regressive Integrated Moving Average): é un modello statistico autoregressivo, in cui le variabili in input e le variabili target appartengono alla stessa serie di dati. Nella versione piú semplice, il modello non supporta molteplici variabili (primo limite), pertanto abbiamo semplificato il set di dati in una serie temporale univariata (altezza del flusso media): considerati N frames $0 \rightarrow i$, μ_{i-1} sono gli input, μ_i é il target.
- **Task 1 - Convolutional Auto Encoder**: considerati i limiti e le perfomance del modello ARIMA, abbiamo affrontato il problema della *next frame prediction* con una *Convolutional Neural Network*. L'input é processato da piú blocchi convoluzionali inframezzati da livelli *Average Pooling* e *Batch Normalization*.

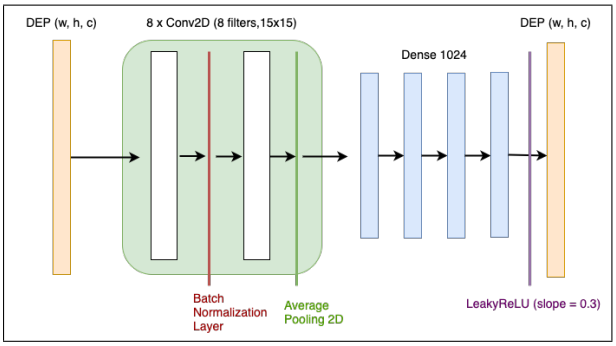

Figure 7. Convolutional Autoencoder (CNN + Dense)

I blocchi convoluzionali costituiscono la parte **encoder**, seguita da un **decoder** costituito da strati densi di dimensione variabile. Abbiamo testato anche un decoder costituito da livelli *Deconvoluzionali 2D* ad un elevato costo computazionale, lasciando questo caso a future considerazioni.

- **Task 2 - Classificatore binario**: come illustrato nella **Figura 6**, abbiamo impostato una rete feed forward che processi la batimetria **BTM**, concatenata con la portata breccia **B**; lo scopo é quello di classificare ogni cella (allagata/non allagata) in base al terreno. In questo caso é applicata la funzione **sigmoid** sul layer finale:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

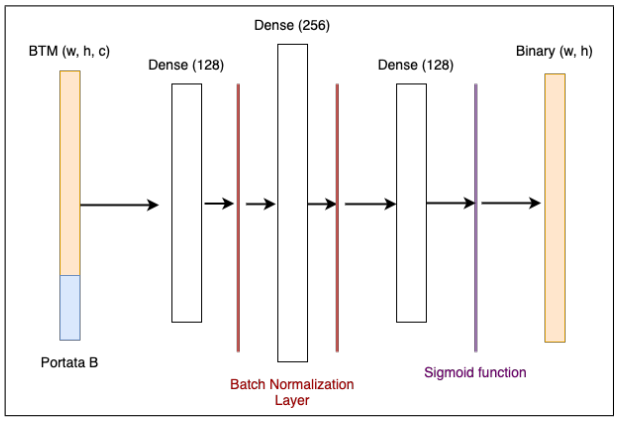

Figure 8. Autoencoder Feed Forward

- **Task 3 - Previsione frame** Δ**T**: rispetto all'architettura del **Task 1**, si combinano due branch per input: un ramo convoluzionale per la batimetria **BTM**; un ramo con rete ricorrente (LSTM) per la portata breccia **B**. Si applica infine una semplice concatenazione, passando per piú strati densi.

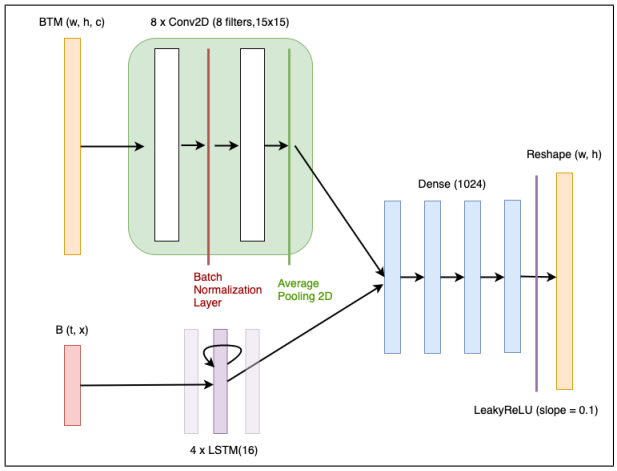

Figure 9. Autoencoder CNN + LSTM

8. EVALUATION

Per la valutazione dei modelli abbiamo utilizzato due principali metriche:

- Keras Accuracy:

$$K(\widehat{Y}, Y) = \frac{(Y - \widehat{Y})}{Y} * 100 \tag{5}$$

- Cosine Similarity:

$$Similarity(\widehat{Y}, Y) = \frac{(Y \cdot \widehat{Y})}{||\widehat{Y}|| * ||Y||} \qquad (6)$$

Task	Modello	Metrica	Score
1	*ARIMA*	Keras Accuracy	72,5%
1	*CNN Autoencoder*	Keras Accuracy	18,2%
2	*Feed Forward*$_1$	Cosine Similarity	70,9%
2	*Feed Forward*$_2$	Cosine Similarity	62,8%
3	*CNN + LSTM*	Keras Accuracy	71,9%
3	*CNN + LSTM*	Cosine Similarity	**92.0%**

[1] risoluzione 33%, 3 layer 128x256x128
[2] risoluzione 100%, 3 layer 512x1024x512

La fase di *Hyperparameters Optimization* é stata effettuata utilizzando il framework **Weights Biases**, il quale testa combinazioni possibili di parametri (**Grid Search**).

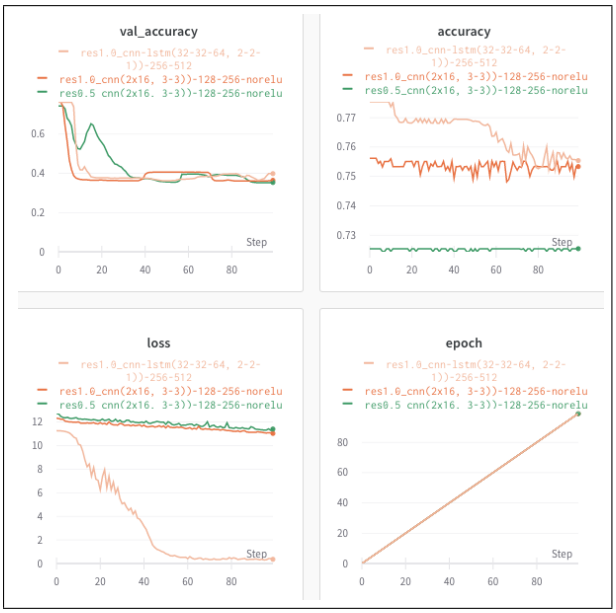

Figure 10. Confronto tra reti CNN+LSTM in fase di Fine Tuning (WeightsBiases)

9. CONSIDERAZIONI

- **Problema Next Frame Prediction**, i modelli scelti presentano due grosse limitazioni: ARIMA é un modello univariato che non puó processare immagini, pertanto é insufficiente per questo studio; le performance del CNN Autoencoder sono scarse, la rete non é in grado di generare un numero di pixels sufficiente per il livello di dettaglio richiesto. La soluzione potrebbe essere quindi passare ad un'architettura piú complessa o del tutto differente.

- **Transformers e Vision Transformers**: le reti CNN, come anche le reti LSTM, sono state progettate in funzione del tipo dei dati da processare (es: le convoluzioni scorrono dei filtri su matrici, aspettandosi quindi delle immagini), pertanto é presente un *Bias Induttivo*. Eliminare la presenza di un bias induttivo potrebbe forse svincolare il modello dalle limitazioni riscontrate. I Transformers sono un nuovo modello computazionale generico: simile nel principio alle reti feed forward, le connessioni nei layer interni sono calcolate "dinamicamente", ed é introdotto il concetto di *Attenzione* sui valori di input (learnable). Recenti paper di ricerca dimostrano la possibilitá di applicare questa architettura alla classificazione e generazione di immagini [6].

- **Modelli non generativi**: alternativamente alla generazione del frame futuro, si potrebbe addestrare un modello ad apprendere soltanto le variazioni tra pixel in due step temporali. In tal caso la matrice in input si somma con il risultato del modello, un principio analogo alle **Residual Networks** e le connessioni [5].

- **Frame ΔT**: considerati i limiti di complessitá della next-frame prediction, abbiamo ridotto il problema alla previsione di un frame nel futuro a distanza ΔT. Il modello Autoencoder CNN + LSTM risulta essere il miglior modello testato: processa frame a risoluzione nativa e genera ad alto livello di dettaglio una matrice di profonditá di fluido con valori continui. La funzione di attivazione piú opportuna é risultata essere la *LeakyReLU*, con un negative slope di 0.3/0.5. Infine, l'aumento del numero di neuroni negli strati densi (2048) ha permesso di aumentare il numero di pixel dettagliati nella previsione, eliminando dei "glitch" nell'immagine.

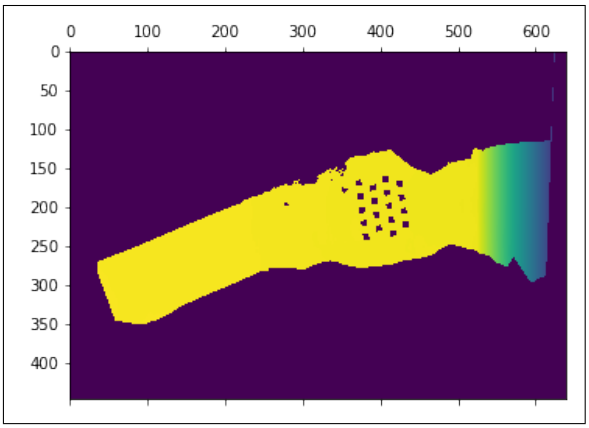

Figure 11. Frame ΔT atteso.

In conclusione, le Simulazioni Shallow Water trovano possibili soluzioni con il Machine Learning. Attraverso misurazioni real-time di un bacino idrico, l'obbiettivo é prevedere il suo stato di allagamento nel futuro.
Le serie temporali semplici sono una potenziale soluzione, tuttavia non risultano sufficientemente scalabili per il numero di variabili considerate.
Le reti neurali offrono strutture flessibili per feature multidimensionali e temporali, ulteriori studi seguiranno per investigare nuove architetture e possibilitá di previsione.

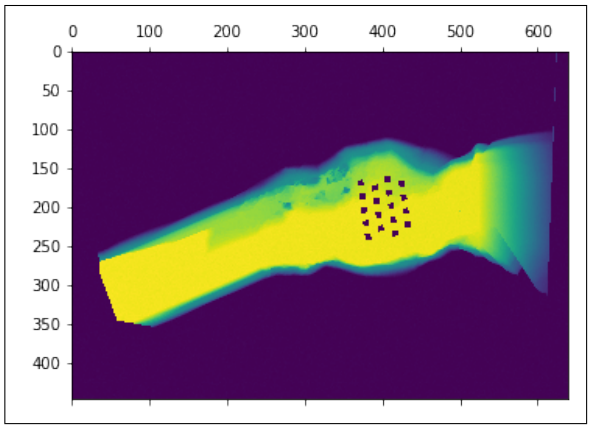

Figure 12. Frame ΔT generato dal modello CNN+LSTM finale.

10. REFERENCES

[1] *GPU-enhanced Finite Volume Shallow Water solver for fast flood simulations*.
R. Vacondio, A. Dal Palù, P. Mignosa (2014)
http://bit.ly/3q4T1H1

[2] *Apprendimento e predizione di dati fluviali tramite reti neurali*.
M. Varesi (2018)
http://bit.ly/3rNGtEv

[3] *A deep convolutional neural network model for rapid prediction of fluvial flood inundation*.
Syed Rezwan Kabir, S. Patidar, Xilin Xia, Qiuhua Liang (2020)
https://bit.ly/3cROWlG

[4] *Simulazioni ad alta risoluzione degli allagamenti-generati da brecce arginali in zone urban*.
F. Vignali (2020)
http://bit.ly/3rNGtEv

[5] *Generating the Future with Adversarial Transformers*.
C. Vondrick, A. Torralba (2020)
https://bit.ly/3p89GIr

[6] *An image is worth 16x16 words: Transformers for Image Recognition at scale*.
A. Dosovitskiy, N. Houlsby (2020)
https://bit.ly/3rJpIu1

[7] *Attention is All You Need*.
A. Vaswani (2017)
https://bit.ly/3q7MueC

[8] *Tensorflow.org Documentation*.
https://www.tensorflow.org/api_docs

[9] *Keras.org Documentation*.
https://keras.io/api/

[10] *PyTorch.org Documentation*.
https://pytorch.org/docs/stable/

[11] *Pierian Data - Education*.
Data Science Bootcamp
Neural Networks with Tensorflow Bootcamp
https://www.pieriandata.com/

[12] *Weights & Biases Documentation*.
https://github.com/wandb

[13] *Equazioni di Navier Stokes - Wikipedia*.
http://bit.ly/36ZZZFF

[14] *Laboratorio HyLab - Modello PARFLOOD*.
http://bit.ly/3tH7VoZ

[15] *ARIMA Model – Complete Guide to Time Series Forecasting in Python*
Marchinelearningplus.org
http://bit.ly/3tJB3vS

[16] *ARIMA Model Python Example — Time Series Forecasting*.
Towardsdatascience.com
http://bit.ly/3q8nK62