UKAN Weeder

Individuazione di piante infestanti tramite UKAN

Agricoltura di precisione

- Monitoraggio in **tempo reale** delle colture per razionare in modo dinamico le risorse quali acqua per l'irrigazione, pesticidi etc.
- Droni impiegati per l'ispezione di vaste aree coltivate: essi dispongono di risorse di **memoria limitate**

Il task su cui lavoriamo consiste nell'individuazione di piante infestanti nelle foto scattate dai droni, per sondare nel **minimo tempo** possibile le piantagioni e dosare il pesticida erogato a ciascuna area: precedentemente il lavoro che viene svolto da un drone il poche ore veniva effettuato nel giro di giorni o settimane da operai umani;

Il task

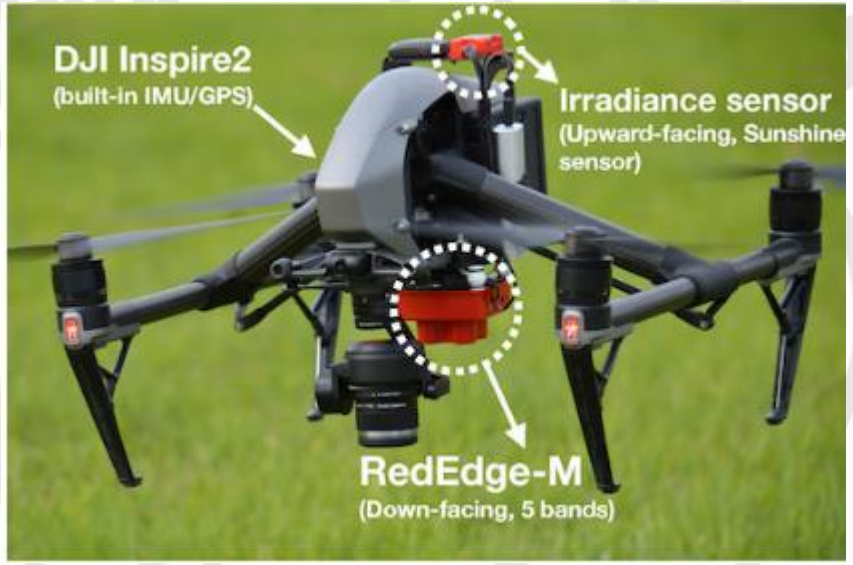
- Da un punto di vista tecnico il task consiste, quindi, nella **segmentazione semantica** a tre **classi esclusive** (background, crop, weed) dei pixel nelle immagini dei campi
- In questo ambito il Deep Learning si è rivelato più efficace rispetto a tecniche di machine learning classico quali Random Forest o SVM: in particolare è stata utilizzata con successo **U-Net** sulle immagini multispettrali del dataset Weedmap
- Risultati di ricerca dimostrano che modelli di Deep learning contenenti **strati KAN**, si rivelano vincenti nella segmentazione semantica di immagini per il **Remote Sensing** rispetto all'attuale stato dell'arte

Dal momento in cui le immagini dei droni hanno caratteristiche simili a quelle satellitari l'obiettivo è capire se si possono ottenere dei buoni risultati anche nell'agricoltura di precisione utilizzando **strati KAN** in una **U-Net**.


Dataset

Weedmap

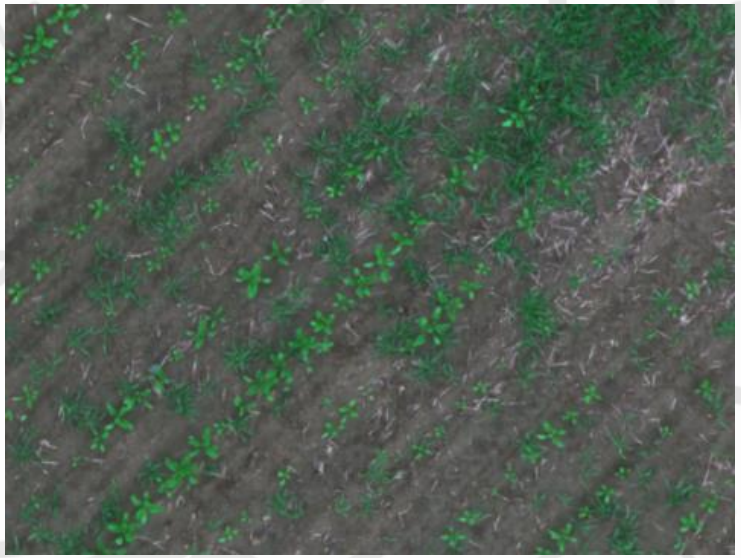
- Il dataset utilizzato nello studio è **Weedmap**: usato come **benchmark**, contiene immagini multispettrali di 8 campi coltivati in Svizzera e Germania, raccolte da due droni: RedEdge e Sequoia.
- Lo studio si focalizza, però, solo sui canali non multispettrali: si usano, quindi, solo i **canali RGB** forniti da **RedEdge**; questa fotocamera ha catturato solo immagini dei **primi quattro campi**, quindi si potrà lavorare solo su quelle

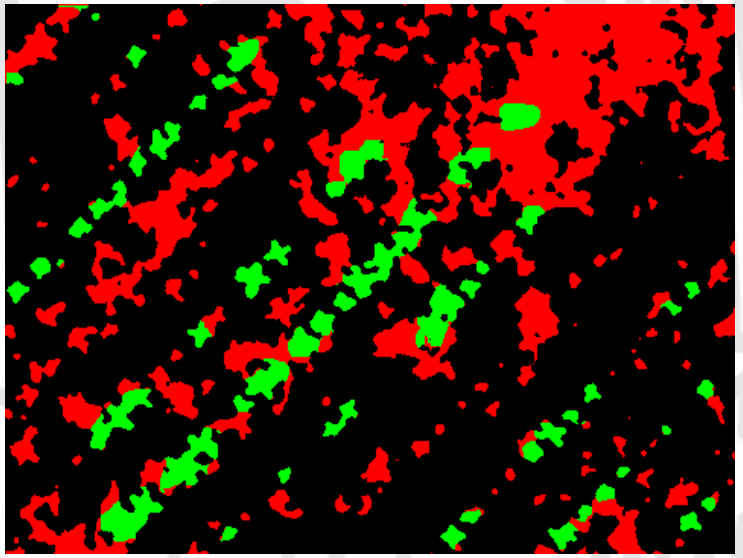


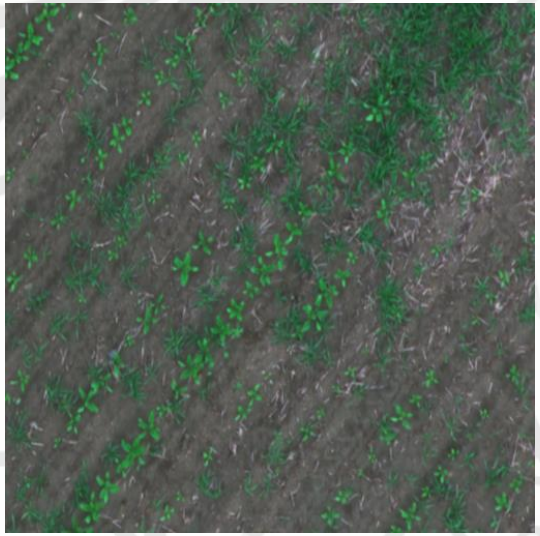
Preprocessing

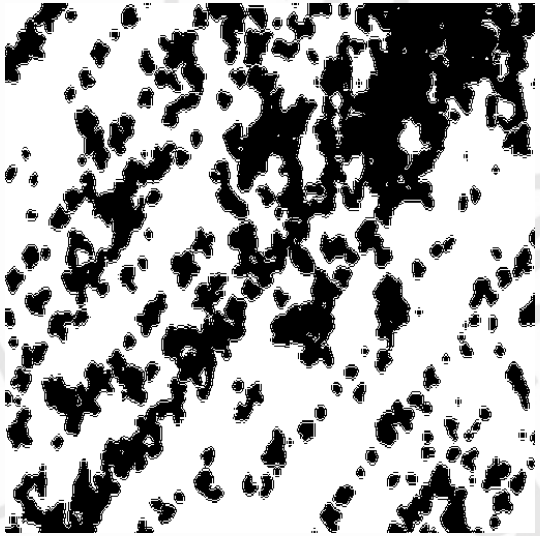
- Per ogni campo mappa ortomosaica -> tiles 360 x 480 pixel -> per compatibilità con studi precedenti **resizing** delle tiles a 512 x 512 pixel
- Creazione **maschere per ogni classe** a partire dalla groundtruth per questioni di compatibilità con il framework utilizzato dal modello
- **Eliminazione foto con solo background** dal dataset di training (campi 000, 001, 002, 004) per limitare la sovrarappresentazione dei pixel di classe background

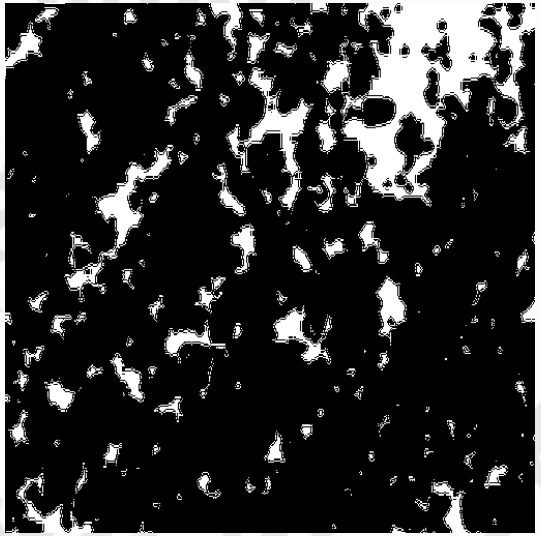
Preprocessing

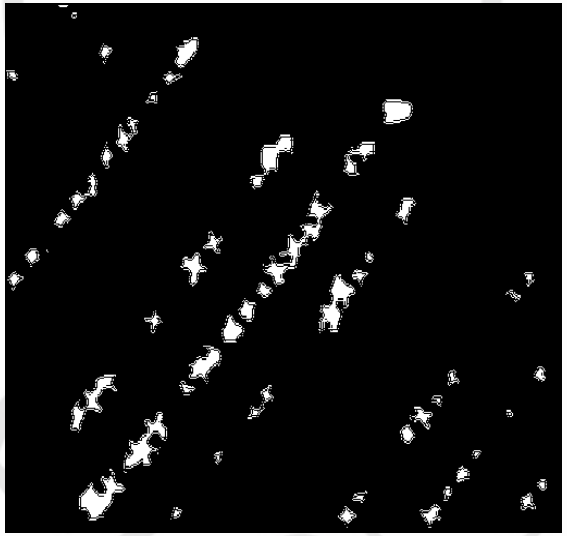












Modello

Modello scelto

- Il modello utilizzato per l'esperimento è UKAN: integrazione di **U-Net** con **strati KAN**
- La sua implementazione è disponibile al seguente link: https://github.com/CUHK-AIM-Group/U-KAN
- testato su dataset di **tipo medico** le performance si sono rivelate **migliori** rispetto alla tradizionale **U-Net**, persino ad un costo **computazionale** inferiore

UNeXt

- Nell’ambito della segmentazione semantica è già stato proposto il modello UNeXt: contiene negli ultimi strati dell’encoder e nei primi del decoder dei **blocchi MLP**
- in questo modo si può riuscire a catturare l’informazione **complessa** contenuta in **tutta la featuremap** estratta dall’encoder, a discapito di quella inerentemente spaziale; spazialità che a livelli di elaborazione molto avanzata è comunque meno presente

Da UNeXt a UKAN

- UKAN sostituisce i **blocchi MLP** con i **blocchi KAN** e ciò potrebbe indirizzare verso delle performance ulteriormente migliorative
- le reti KAN apprendono **non** i pesi di **combinazioni lineari**, che vengono poi attivate, ma **direttamente** di **funzioni** più **complesse**: intuitivamente ciò potrebbe conferire alla rete un potere espressivo maggiore
- l'intuizione viene confermata dal **teorema di Kolmogorov-Arnold**

Da UNeXt a UKAN: Kolmogorov-Arnold

Secondo il teorema di Kolmogorov-Arnold, una funzione continua multivariata può essere riscritta come combinazione di un numero finito di funzioni continue univariate secondo la seguente formula

$$f(x) = \sum_{q} \phi_q \left(\sum_{p} \psi_{pq}(x_p) \right)$$

le funzioni ϕ_q e ψ_{pq} sono **spline-functions** di cui vanno appresi i **parametri**, ma di solito $\phi_q = I$.

- Si tratta di un risultato teorico più confortante del teorema di approssimazione universale che sta alla base degli strati MLP: se uno strato KAN può in **potenza** imparare la **funzione esatta**, un MLP può solo **approssimarla**.

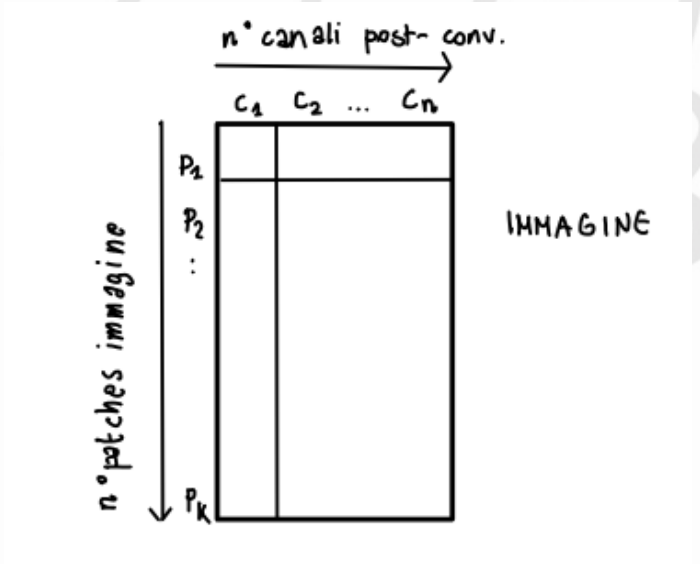
Architettura di UKAN

- Come U-Net: di tipo encoder-decoder del tutto **speculare** e con **skip-connections** tra i corrispondenti livelli di encoder e decoder
- Ogni **livello** dell'encoder **dimezza** altezza e larghezza della immagine, mentre i **canali risultanti** sono degli **iperparametri**
- per i **primi tre livelli**, che sono **convoluzionali**, si indica solo il numero di **canali** per il **primo livello**, gli altri due sono un quarto ed un ottavo di questo numero.
- Gli **ultimi due livelli** sono **Tokenized-KAN block** e sono composti da 3 moduli:
  - Modulo di Tokenizzazione: divide le immagini in token lineari, da poter passare ai layer KAN
  - Modulo KAN: formato da **tre layers** prende in input la matrice ricavata dalla tokenizzazione e la trasforma senza variarne però alcuna dimensione
  - altri due moduli che prendono in input le immagini **delinearizzate** e **non** ne **cambiano** le **dimensioni** (si tratta di una convoluzione depth-wise, una batch-normalization e l'attivazione tramite ReLU)

Modulo di Tokenizzazione

- **Convoluzione**:
  - kernel = 3, dimensione di una patch/token
  - Canali = iperparametro

Linearizzazione



- **Concatenazione** sulle righe, delle matrici provenienti dalle immagini del batch
- Per **delinearizzare** l'immagine vengono **conservate**:
  - altezza e larghezza delle immagini, dopo la convoluzione e prima della linearizzazione
  - dimensione del batch, prima della concatenazione

Sperimentazione

Setup della sperimentazione

- le features estratte dal modello venivano utilizzate per task di classificazione multilabel; si è definita, invece, una **Cross Entropy**, funzione di perdita per classi esclusive
- Campi 000, 001, 002, 004 **training** set; 003 **test** set
- Parametri di addestramento di default:
  - Ottimizzatore **Adam**: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda = 1e-4$
  - **Cosine annealing**: learning rate che varia in funzione delle epoche in un intervallo predefinito
    - Strati **non KAN**: [1e-4 e 1e-5]
    - Strati **KAN**: [1e-2 e 1e-5]
    - 400 epoche
    - Batch size = 8
- Google Colab, GPU T4

Modelli confrontati

- Segformer **MiT-b0**:
  - Versione con **meno parametri** dato l'intento perseguito di fare segmentazione a basso costo computazionale
  - **Fine tuning** di 200 epoche, con batch size di 8, del modello preaddestrato di Hugging Face
- UKAN **3 configurazioni** di **iperparametri** 'input list':
  - Default [128, 160, 250]
  - Diminuire la complessità del modello, mantenendo le **potenze di 2** le **proporzioni** tra gli elementi della lista: [64, 80, 128], [32, 40, 64]

Metriche utilizzate

- Per **l'accuratezza** si è usata la **F1-score**, per questioni di **compatibilità** con studi precedentemente condotti sullo stesso topic: in effetti si tratta di una metrica resistente allo **sbilanciamento** tra classi
- La **complessità** è valutata in termini di tre diverse metriche riguardanti la **grandezza** del modello e le sue **tempistiche**:
  - Numero di **parametri**
  - Numero di **operazioni** svolte dal modello (in GMACs)
  - **Tempo** impiegato per svolgere la **predizione** (inference time)

Risultati

Tabella 1: Risultati sperimentazione

		F1					
Metodo	F1-avg	BG	Coltivazioni	Infestanti	GMACs	Paramtetri (M)	Inference Time (ms)
UKAN-128	71,5	**99,4**	**55,7**	59,3	7,07	6,36	18
UKAN-64	**72,1**	**99,4**	55,2	**61,8**	**1,84**	**1,60**	18
UKAN-32	33,1	99,3	0,0	0,0	/	/	/
Segformer (MiT-b0)	69,5	**99,4**	50,1	58,9	7,84	3,71	**13**

- Il **test set** = campo 003
- **Batch size** per l'**inferenza** era di 8 immagini

Risultati

- i modelli UKAN detengono le migliori performance in termini di **F1-score**:
  - UKAN-64 ha la migliore **accuratezza** e i migliori indicatori di **grandezza**: pare, infatti, essere il **compromesso** più adatto tra l'eccessiva complessità di UKAN-128 e l'underfitting di UKAN-32
- Segformer ha **tempistiche** di inferenza di **pochi millesimi** migliori

Il risultato si ritiene soddisfacente in quanto UKAN-64 risulta **più precisa** e **meno grande**, in termini di parametri ed operazioni, rispetto alla più leggera versione di SegFormer.

Future direzioni di ricerca

Dal momento in cui l'utilizzo di UKAN-64 pare essere una scelta vincente, future direzioni di ricerca potrebbero consistere nell'**integrazione** di tale archiettura in modelli **non supervisionati**, come **Roweeder**.


Grazie