



Studenti:

ANTONUCCI GAETANO matr. 0522500941

PAGLIARA NICOLA matr. 0522501413

Università degli Studi di Salerno

Un approccio **deep learning** per la **classificazione** delle varie tipologie di **forme tumorali** tramite uso delle **espressioni genetiche**, individuando potenziali **biomarker**

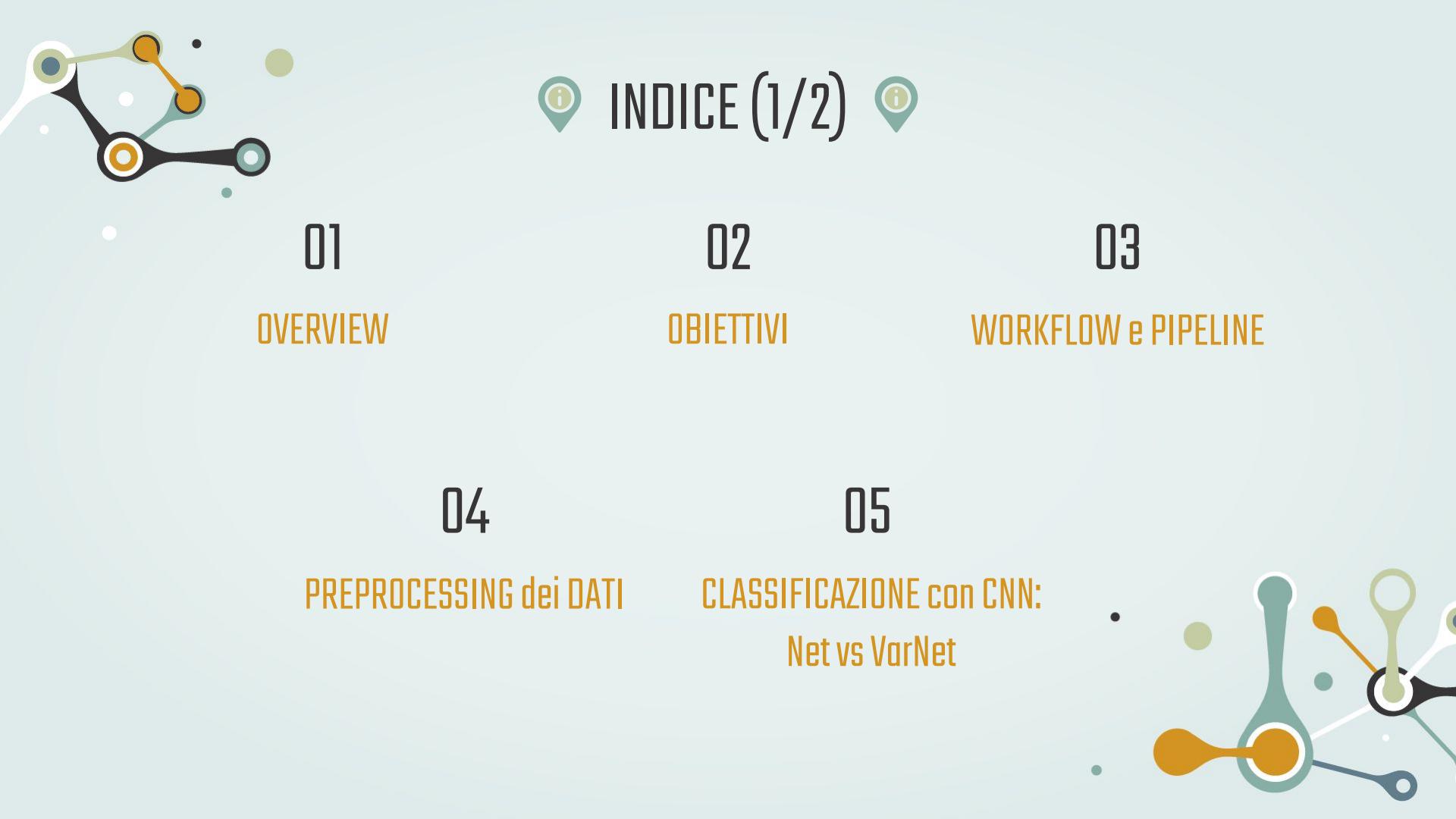
Strumenti Formali per la Bioinformatica

A.A. 2022/2023

Prof. Rocco Zaccagnino

Prof.ssa Clelia De Felice

Prof.ssa Rosalba Zizza



INDICE (1/2)

01

OVERVIEW

02

OBIETTIVI

03

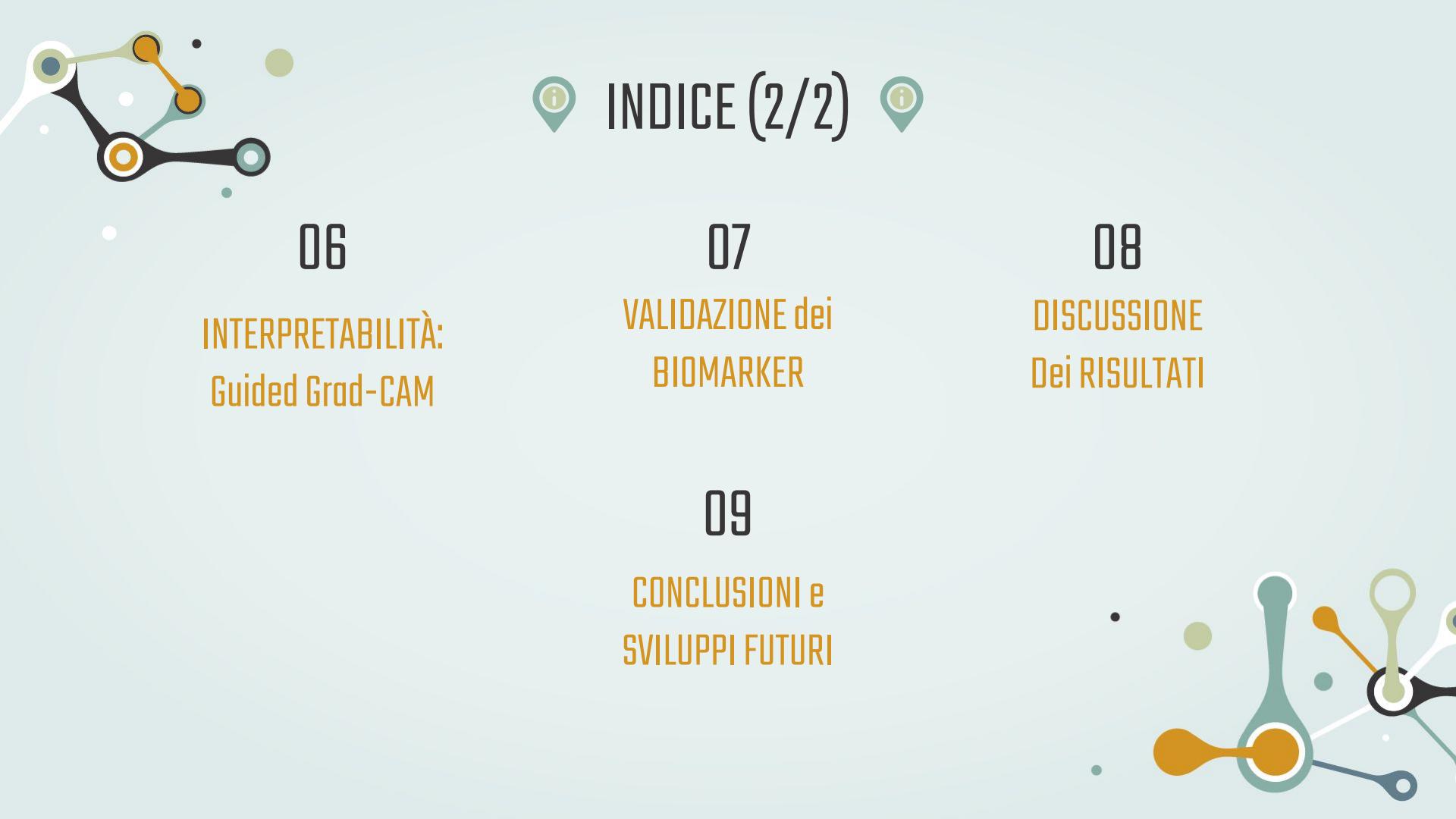
WORKFLOW e PIPELINE

04

PREPROCESSING dei DATI

05

CLASSIFICAZIONE con CNN:
Net vs VarNet



INDICE (2/2)

06
INTERPRETABILITÀ:
Guided Grad-CAM

07
VALIDAZIONE dei
BIOMARKER

09
CONCLUSIONI e
SVILUPPI FUTURI

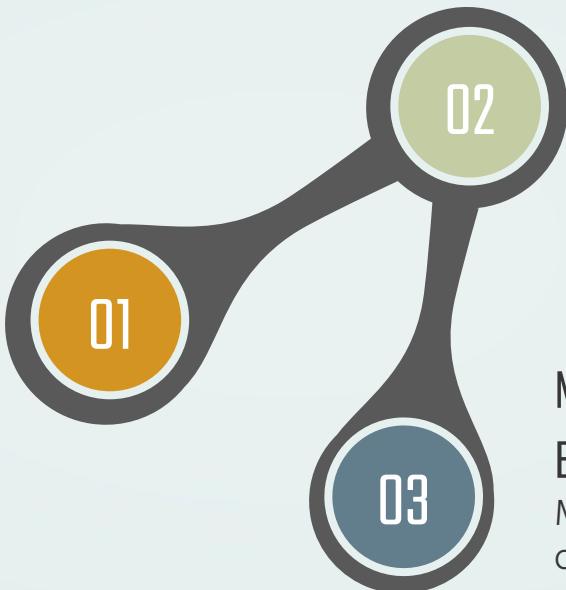
08
DISCUSSIONE
Dei RISULTATI



OVERVIEW

Introduzione all'applicazione e al
contesto in cui opera

HIGHLIGHTS



INTRODUZIONE

Una breve analisi del contesto in cui sussiste la problematica affrontata

LAVORI CORRELATI

Una breve panoramica dei lavori presenti in letteratura

METODOLOGIA APPLICATA

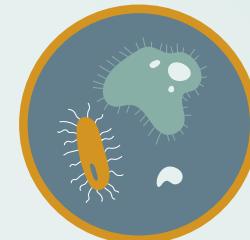
E RISULTATI OTTENUTI

Metodo utilizzato e risultati ottenuti dal progetto

? INTRODUZIONE ?

- Con il **miglioramento** delle **tecniche di sequenziamento** con l'avvento dei Next Generation Sequencing methods (**NGS**), si è avuto un progressivo miglioramento in **accuratezza** ed **efficienza** anche per quanto riguarda l'analisi del **genoma umano**.

UN PROBLEMA SEMPRE PRESENTE



- **comprendere** in maniera completa e approfondita le **cause** dei **vari tumori**
 - sfruttando la **conoscenza** del **genoma umano** per scoprire e mappare le **complessità biologiche** dei tumori utilizzando una **tecnica diversa** da quelle presenti nella letteratura biologica
- e **scoprire** quali sono i **geni responsabili** del loro sviluppo (**biomarker**)
- Per provare a **risolvere** questo **problema**, analizziamo meglio il **progresso** e lo sviluppo raggiunto dalle NGS nel trattare il problema del sequenziamento dell'RNA.



OMICS: RNA-sequencing

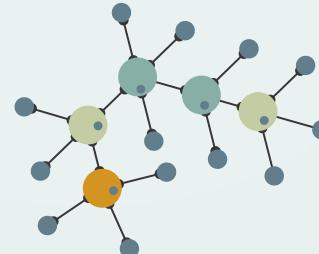
- Con il termine OMICS si intendono tutte quelle scienze biologiche il cui nome termina con il **suffisso -omics**, ad esempio **genomics, transcriptomics, proteomics, metabolomics**.
- Per cercare di dare una soluzione al problema esposto, abbiamo sfruttato il sequenziamento dell'RNA che è un problema affrontato dalla **trascrittomica (studio dell'RNA nelle molecole)**
 - L'**RNA-Seq** conta le **singole sequence read** allineate ad una reference sequence per generare conteggi discreti di read quantificando quindi la presenza di molecole RNA nel campione biologico.
 - In particolare, per la **ricerca sul cancro**, si è rivelato uno **strumento essenziale** per misurare in maniera diretta le **conseguenze funzionali** delle **mutazioni dei geni**.
 - Dalle 46 mutazioni circa contenute nel cancro medio, ne servono solo da 5 a 8 per dare inizio ad un tumore.
 - tramite **RNA-Seq** è possibile **differenziare** in maniera imparziale **i fattori cruciali** per la **progressione del cancro**, combinando le misurazioni dei modelli di espressione genica e le conseguenze delle mutazioni.



RNA-Seq: NGS per il sequenziamento RNA

- La tecnica di RNA-Seq è utilizzata nel campo della trascrittomica, estraendo conoscenza dalla funzione biologica descritta dalle informazioni contenute nel DNA (i geni) che compongono il genoma di una specie.
- Con il termine Next-Generation Sequencing (NGS), si intendono quei metodi di sequenziamento ideati dal 2000 in poi, ossia quelli di seconda generazione che hanno i seguenti vantaggi:

- sono ad alto throughput
- hanno una coverage elevatissima
- sono veloci e poco costosi

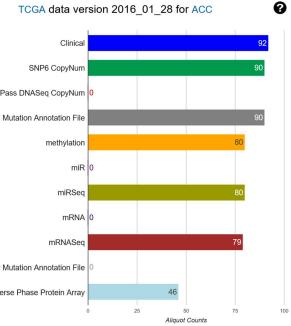


- In questo lavoro saranno sfruttati i dati ottenuti dall'applicazione di RNA-Seq usando le NGS tramite la tecnologia **HiSeq System** di Illumina.



TGCA: Pan-Cancer Atlas

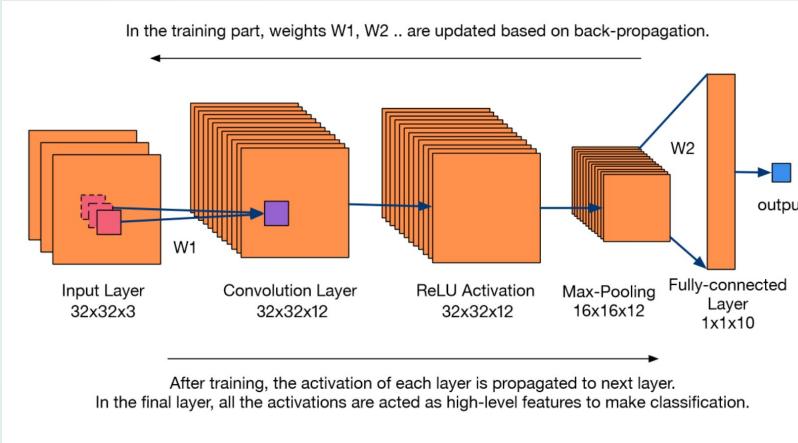
- Fin dall'avvento delle NGS, sono stati raccolti un gran quantitativo di dati di sequenze genomiche
- **The Cancer Genome Atlas**: programma di **riferimento** per la **genomica del cancro**, ha fornito una grande base di conoscenza per comprendere le cause dei tumori
- ha **caratterizzato** molecolarmente oltre **20.000 campioni** primari di cancro e campioni normali appaiati che coprono **33 tipi di cancro**
- Nato nel 2006, in **12 anni**, con i contributi di oltre **11.000 pazienti** e l'incredibile impegno di migliaia di ricercatori, il TCGA ha prodotto una serie di **dati di incommensurabile valore** (si parla di circa **2.5 petabytes**).
- Il TGCA ha stabilito l'importanza della genomica del cancro
- Da questi sforzi è nato poi, il **Pan-Cancer Atlas**, una raccolta di **analisi trasversali** sul cancro:
 - **modelli di origine cellulare**
 - **processi oncogenici**
 - **signaling pathway**



Lavori Correlati (1/2)

Metodi di Deep Learning (DL) per la classificazione

LYU, Boyu;
HAQUE, Anamul.
Deep learning
based tumor type
classification using
gene expression
data. In:
*Proceedings of the
2018 ACM
international
conference on
bioinformatics,
computational
biology, and health
informatics.* 2018.
p. 89-96



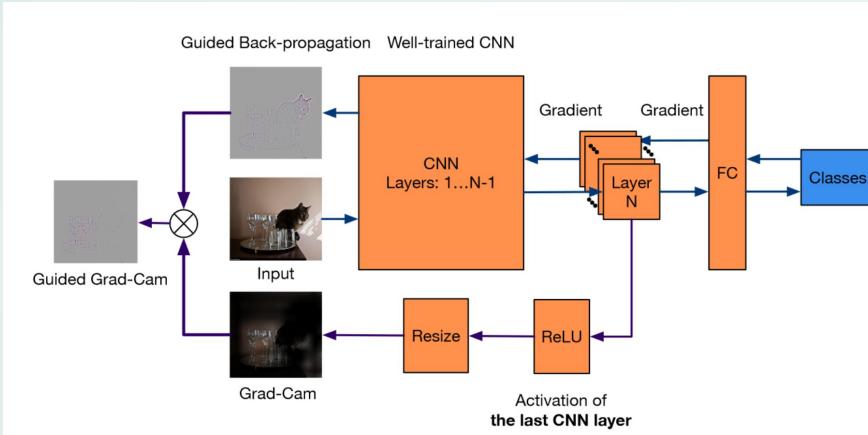
- Molti paper di questi ultimi anni si soffermano maggiormente sull'individuazione dei **top gene** per una **singola tipologia di tumore** mentre altri si limitano ad effettuare una classificazione binaria (identificazione) dei tumori usando i gene expression data.
- Nel lavoro di riferimento, viene scelto il **modello CNN** (Convolutional Neural Network) per classificare le immagini che rappresentano i campioni biologici nelle rispettive tipologie tumorali.

- Le **CNN sono un estrattore di caratteristiche**: più l'immagine andrà in profondità più caratteristiche complesse possono essere individuate.
- La CNN usata nel lavoro di riferimento, a differenza dei paradigmi di ML, **non è soggetta** al problema della “**curse of dimension**” generata dal trattare i dati biologici.
- Tale CNN **assegna ai pixel** di un'immagine (contenenti i geni) uno **score di confidenza** che esplicita la loro contribuzione alla classificazione

Lavori Correlati (2/2)

Metodi di visualizzazione delle Deep Neural Network (DNN)

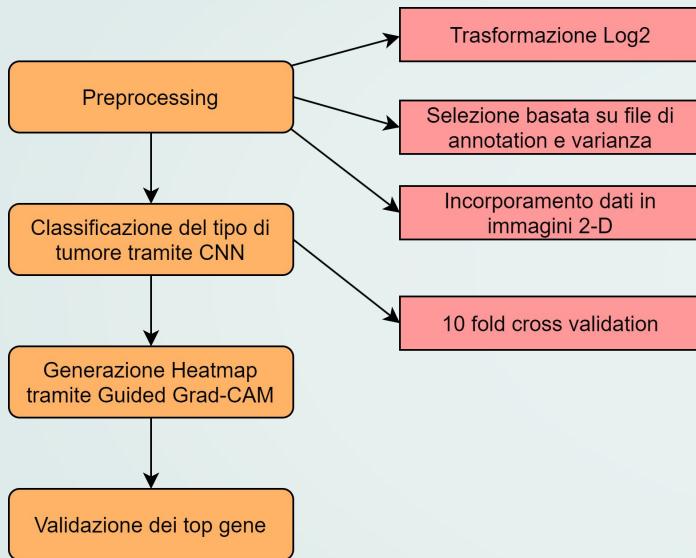
SELVARAJU,
Ramprasaath
R., et al.
Grad-cam:
Visual
explanations
from deep
networks via
gradient-base
d localization.
In:
*Proceedings
of the IEEE
international
conference on
computer
vision. 2017.*
p. 618-626



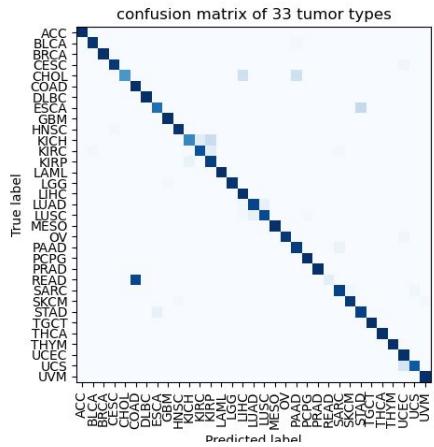
- Per **comprendere** le **decisioni** prese in fase di **classificazione** dal modello, è necessario **visualizzare** come quest'ultimo sia arrivato a compiere tale scelta.
- Al fine di fare ciò, utilizziamo una **heatmap** che deve rispettare le seguenti due proprietà:
 - essere **class-discriminative**
 - ed essere ad **alta risoluzione**

- Una heatmap che rispetta tali proprietà viene generata utilizzando una tecnica chiamata **Guided Grad-CAM**, che è divisa in due parti:
 - a. Generare **Grad-CAM** per capire quali sono le **regioni più importanti** dell'immagine
 - b. Individuare l'**errore commesso** dalla rete in fase di classificazione, per **ogni immagine**, tramite **Guided Backpropagation**.

Metodologia Applicata



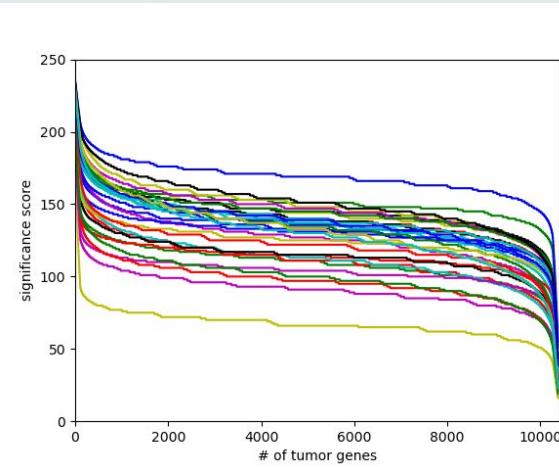
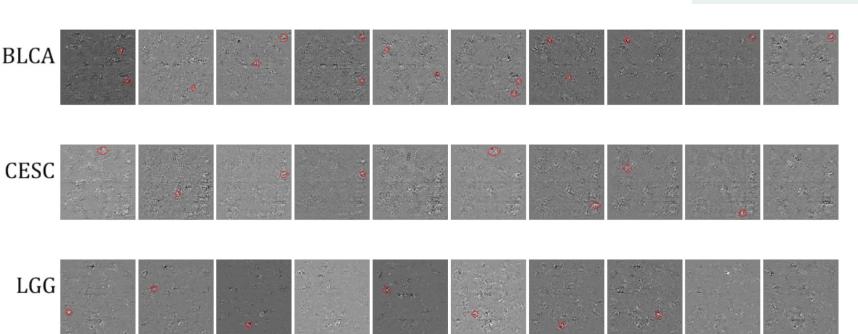
1. I **dati scaricati** vengono sottoposti ad una fase di **preprocessing** dove vengono filtrati e dove sono **scalati** tramite **Log2** per evitare bias ed infine vengono **inglobati** in **immagini** 2D.
2. Il risultato viene dato in pasto alla **rete** che **effettua** la **classificazione**.
3. Tramite i pesi prodotti durante la fase di training, vengono **generate le heatmap** per visualizzare le decisioni prese dal modello.
4. Infine, i risultati del modello, cioè una **lista di geni**, viene **convalidata** come possibili biomarker tramite **Pathway Analysis**.



Risultati Ottenuti

- Esempi degli **artefatti** prodotti che **vedremo** nel seguito.

Metodo	Accuracy	Precision	Recall	F1-score
CNN	95.79%	95.95%	95.79%	95.57%



MESO	hsa04610	Complement and coagulation cascades	2.56e-09
	hsa04080	Neuroactive ligand-receptor interaction	1.26e-07
	hsa04512	ECM-receptor interaction	3.08e-07
	hsa05144	Malaria	6.56e-07
	hsa05150	Staphylococcus aureus infection	7.07e-07
	hsa04020	Calcium signaling pathway	5.55e-06
	hsa04974	Protein digestion and absorption	3.15e-05
	hsa04510	Focal adhesion	3.79e-05
	hsa05205	Proteoglycans in cancer	1.31e-04
	hsa04514	Cell adhesion molecules	1.33e-04
	hsa04390	Hippo signaling pathway	1.33e-04
	hsa04061	Viral protein interaction with cytokine and cytokine receptor	1.34e-04
	hsa04145	Phagosome	1.51e-04
	hsa04915	Estrogen signaling pathway	2.49e-04
	hsa04621	NOD-like receptor signaling pathway	4.07e-04
	hsa05030	Cocaine addiction	5.50e-04
	hsa05165	Human papillomavirus infection	6.73e-04
	hsa05033	Nicotine addiction	6.76e-04
	hsa04015	Rap1 signaling pathway	8.79e-04
	hsa04060	Cytokine-cytokine receptor interaction	9.22e-04
	hsa04350	TGF-beta signaling pathway	9.52e-04

OBIETTIVI

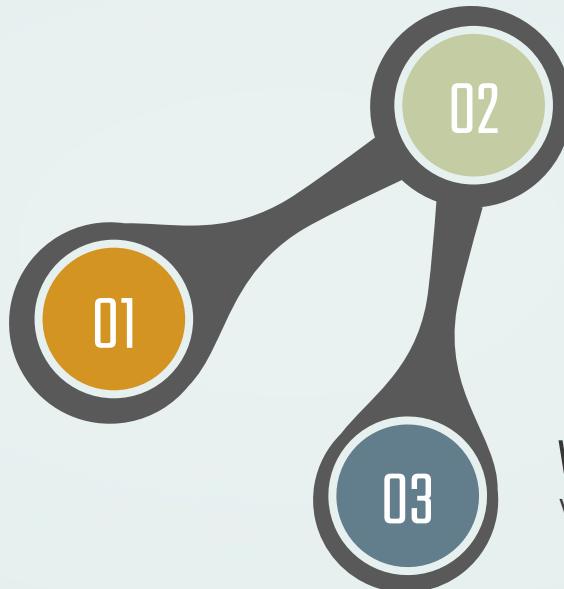
Scopi di dl-tumor-based classification



HIGHLIGHTS

CLASSIFICAZIONE

Classificare le tipologie tumorali



COMPRENSIONE

Analizzare le decisioni prese dal classificatore

VALIDAZIONE

Validare i geni candidati



GOALS dell'Applicazione



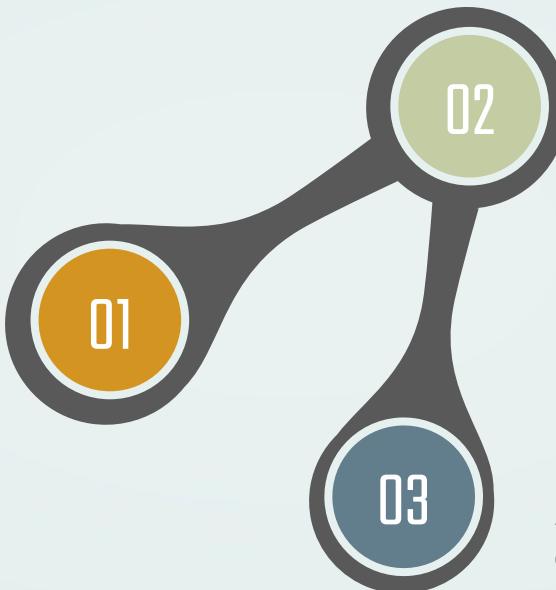
- L'uso di **dl-tumor-based-classification**, che è basato sul lavoro di Boyu Lyu e Anamul Haque "Deep Learning Based Tumor Type Classification Using Gene Expression Data" (In Proceedings of the 2018 ACMinternational conference on bioinformatics, computational biology, and health informatics. 89–96)
si pone i **seguenti obiettivi**:
 - A partire dai **gene expression levels** stimati per ogni campione contenuto nel dataset, **classificare correttamente** a quale coorte tumorale appartiene il suddetto campione.
 - In seguito, **comprendere** come il classificatore è arrivato a prendere tali **decisioni** relative ai sample di ogni coorte tumorale.
 - Una volta compreso quali **geni** hanno **contribuito** di più alla **classificazione**, tali geni sono considerati **candidati biomarker** e verranno convalidati attraverso una tecnica di validazione.
 - Questi geni possono essere dati agli esperti biologi per aiutarli a costruire **sistemi diagnostici più efficaci, veloci ed affidabili**. Oltre questo, tali geni possono essere usati per studiare la **complessità** di alcune **malattie** legate alle varie **coorti tumorali**.

PIPELINE

Refactoring e modularizzazione
del progetto



HIGHLIGHTS



SETUP
Environment
dell'esperimento

PIPELINE
Modularizzazione del
progetto

MODULI
Attività fondamentali svolte
durante l'esecuzione



Setup del workspace (1/2)

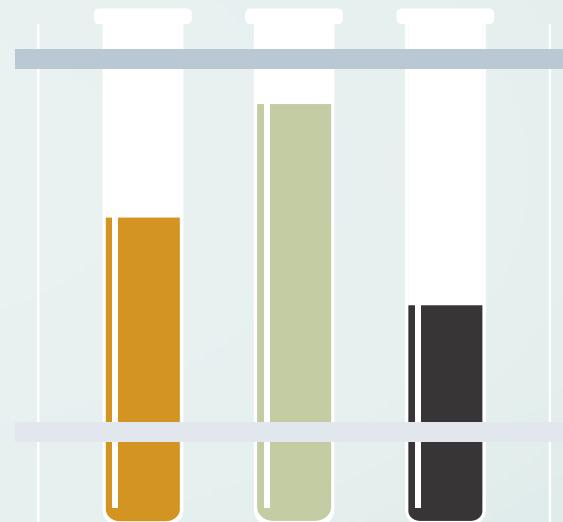


Per effettuare i test abbiamo creato un nuovo environment con Anaconda con il seguente comando:

```
$ conda create -n pydlhh python=3.9
```

E abbiamo installato i seguenti pacchetti:

- numpy (1.23.5)
- pandas (1.5.2)
- pytorch (1.13.1) con CUDA (ver. 11.6)
- matplotlib (3.5.3)
- scikit-learn (1.2.1)
- scikit-image (0.19.3)
- imbalanced-learn (0.10.1)
- opencv (4.7.0)



Setup del workspace (2/2)

I test sono stati effettuati su un **server HP** con le specifiche riportate in tabella:



Come **IDE** abbiamo utilizzato **PyCharm Community Edition**.

Scheda Madre

HPE ProLiant ML350 Gen10

Processore

Intel(R) Xeon(R) Gold 5218 CPU @ 2.30Ghz (x32)

RAM

270 GB

Scheda Grafica

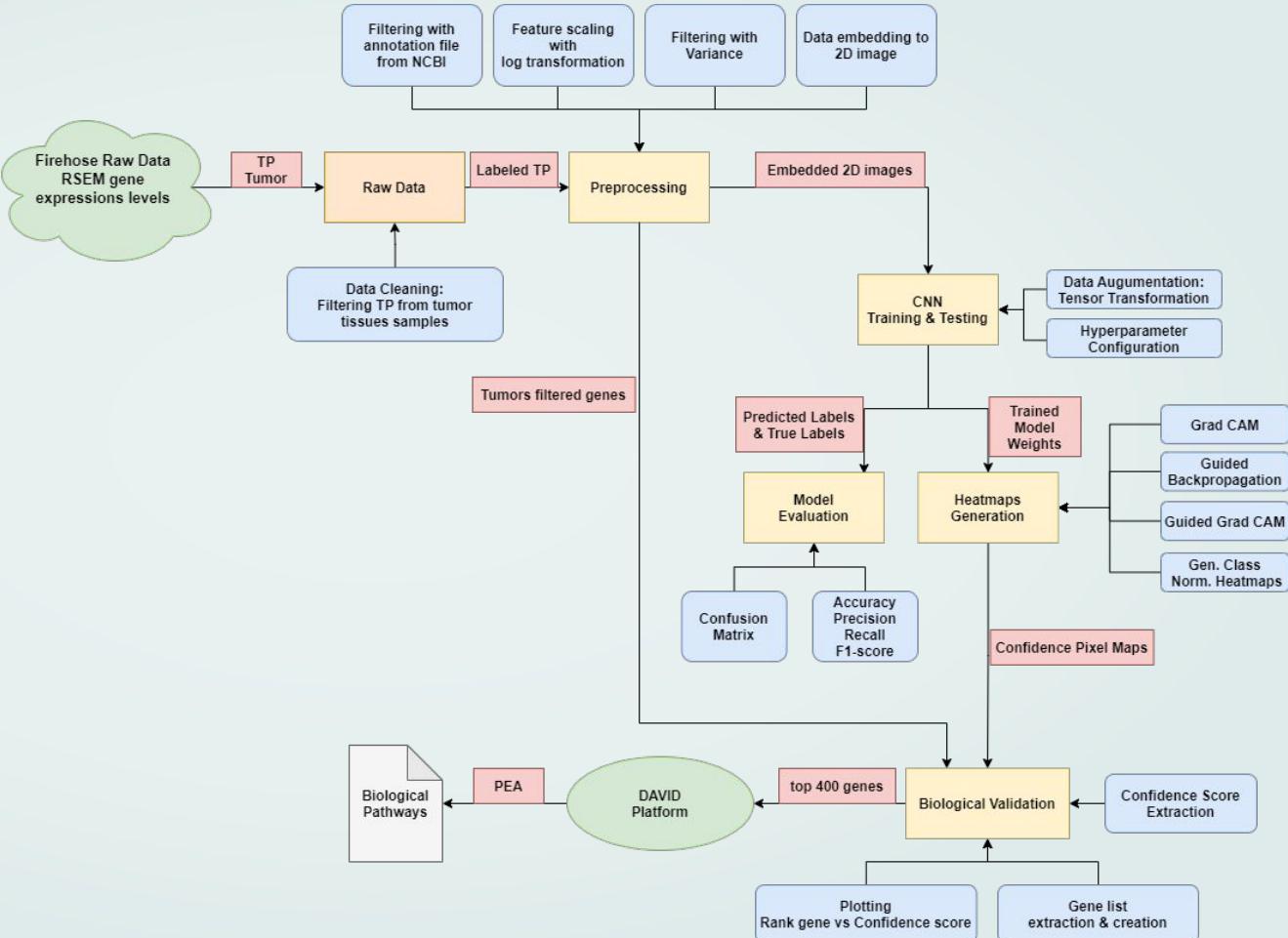
NVIDIA Quadro RTX 4000 (8 GiB di memoria dedicata) [CUDA ver. 11.6]

Sistema Operativo

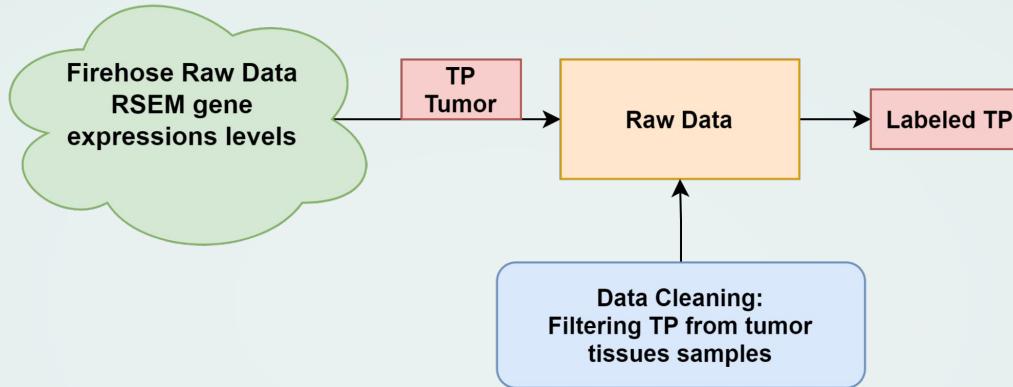
Ubuntu 21.10 (64-bit)

Pipeline

- Come è possibile notare dai riquadri in giallo nell'immagine, la nostra pipeline è divisa in 6 moduli
- I riquadri in celeste indicano i servizi offerti dal modulo
- I riquadri in rosso sulle frecce indicano gli output dei vari moduli
- In verde sono indicati i servizi esterni utilizzati

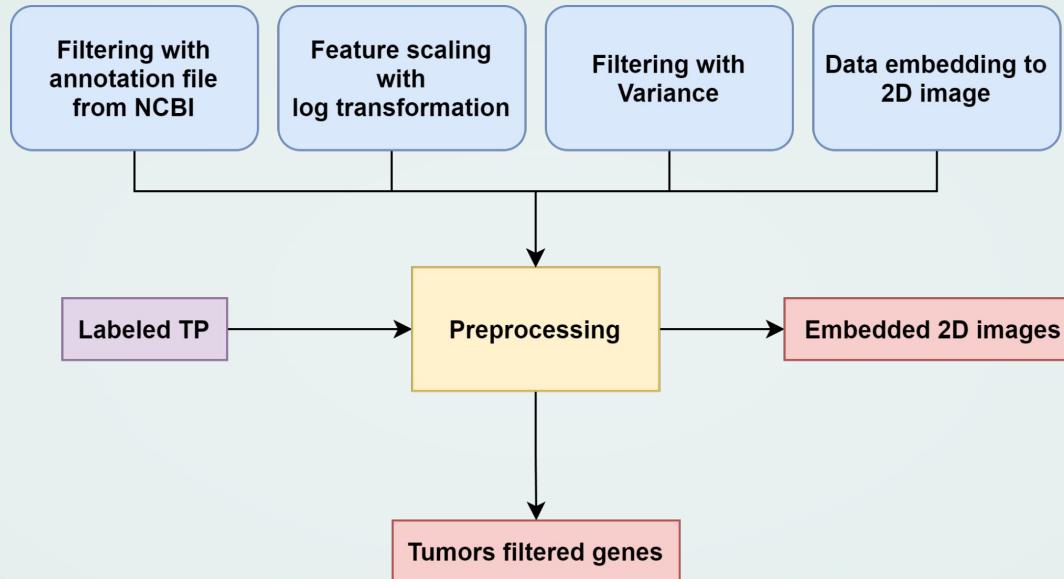


Modulo 0: Raw Dataset



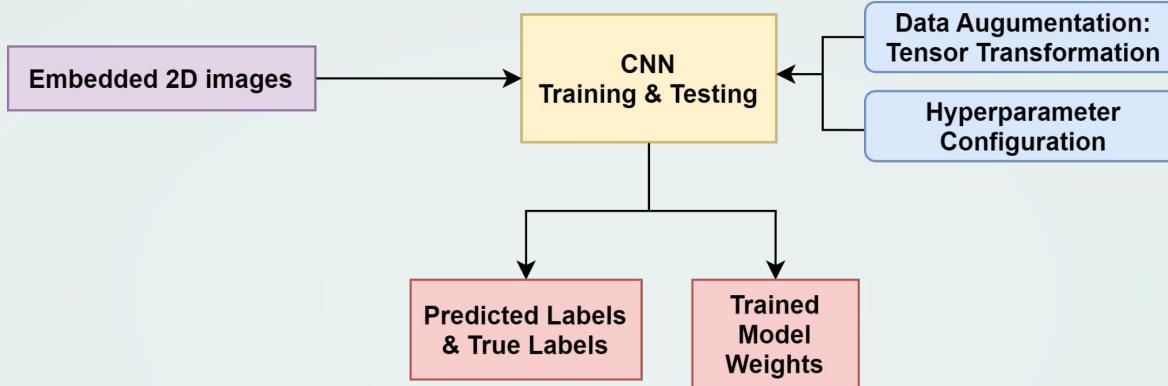
Obiettivo: trattare i dati di **output** di **RSEM**, preventivamente **ripuliti** ed **etichettati**, al fine di generare il dataset utilizzato, unendo i sample delle varie coorti tumorali in un unico file csv.

Modulo 1: Preprocessing



Obiettivo: generare le **immagini 2D** da dare in pasto alla **rete neurale** e generare il **feature set** necessario alla fase di **validazione**

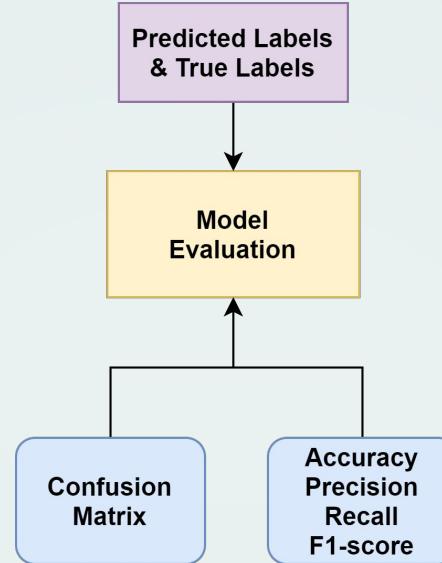
Modulo 2: Training & Testing



Obiettivo: usando le **immagini** generate:

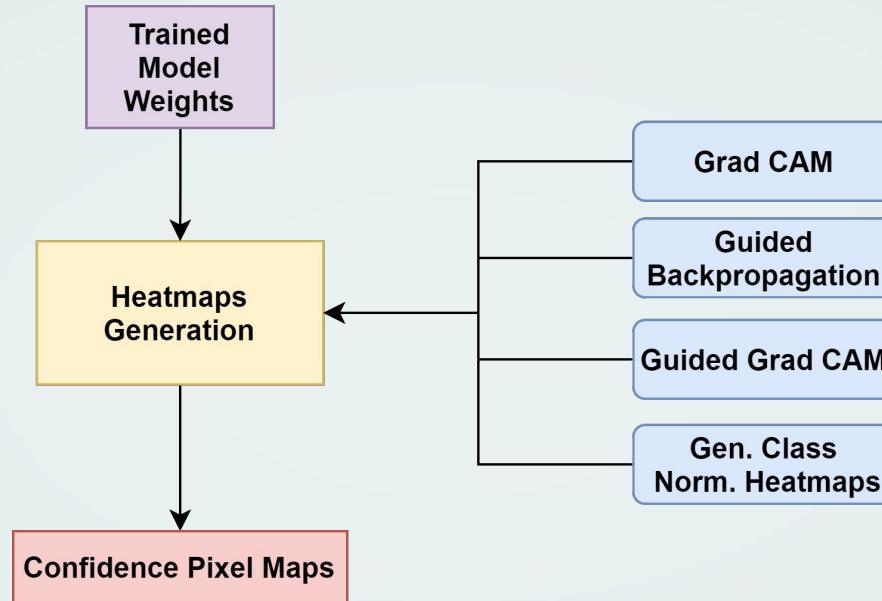
- crea e imposta i parametri necessari per la fase di **training** e di **testing** del modello
- esegue training usando la **10-fold cross validation** ed il testing su ogni fold
- salva i **pesi** ottenuti

Modulo 3: Valutazione delle Performance



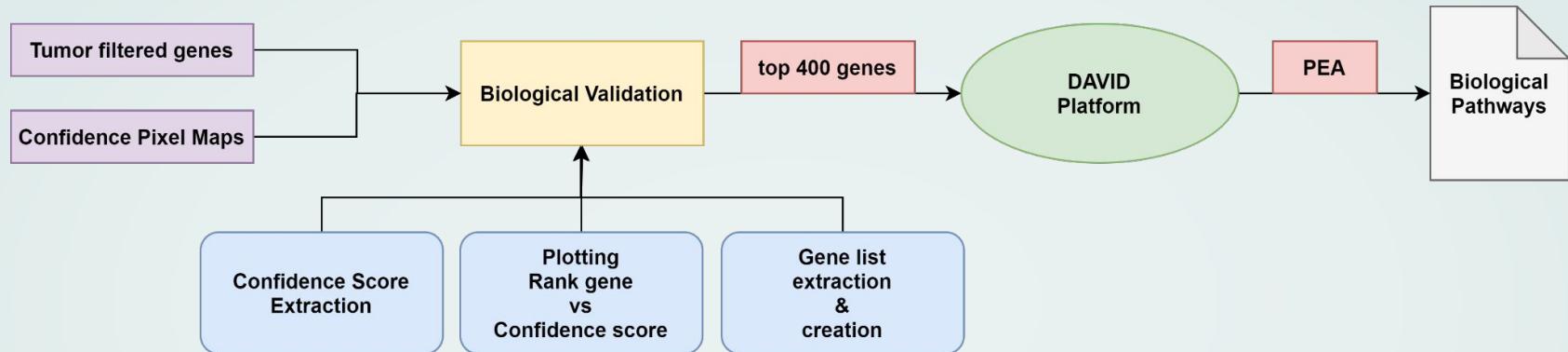
Obiettivo: valutare in maniera **visiva** e **matematica** le **performance** del modello

Modulo 4: Generazione delle Heatmap



Obiettivo: utilizzando i **pesi** del modello, genera le **heatmap** (risultato di **Guided Grad-CAM**) per ogni coorte tumorale

Modulo 5: Validazione Biologica



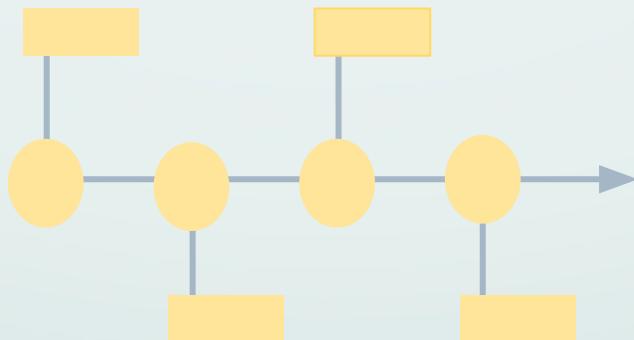
Obiettivo: generare le **liste di geni** ottenuti dalle **heatmap** da sottomettere al tool **DAVID** e **correlarle** a biological pathway relative alle coorti tumorali

Hotspots della Pipeline

Le **attività fondamentali** della metodologia applicata descritte nella sezione 1, che **contribuiscono** in maggior parte agli **obiettivi fissati** nella sezione 2, sono **mappate** dai seguenti **moduli**:

- **Raw data + preprocessing**
- **Training & Testing + Model Evaluation**
- **Heatmaps generation**
- **Biological Validation**

pertanto sono illustrate nel seguito, in maniera più approfondita e dettagliata, tali attività descritte dai moduli presentati.





PREPROCESSING dei DATI

Pre-elaborazione dei dati da
passare al modello



HIGHLIGHTS

RNA-Seq RAW DATA

I dati grezzi scaricati dal progetto [FireHose Broad GDAC](#)

OVERSAMPLING

Utilizzo di SVMSMOTE per bilanciare il dataset

01

02

03

04

FEATURE SCALING, FILTERING and ORDERING

Attività di pre-elaborazione dei dati

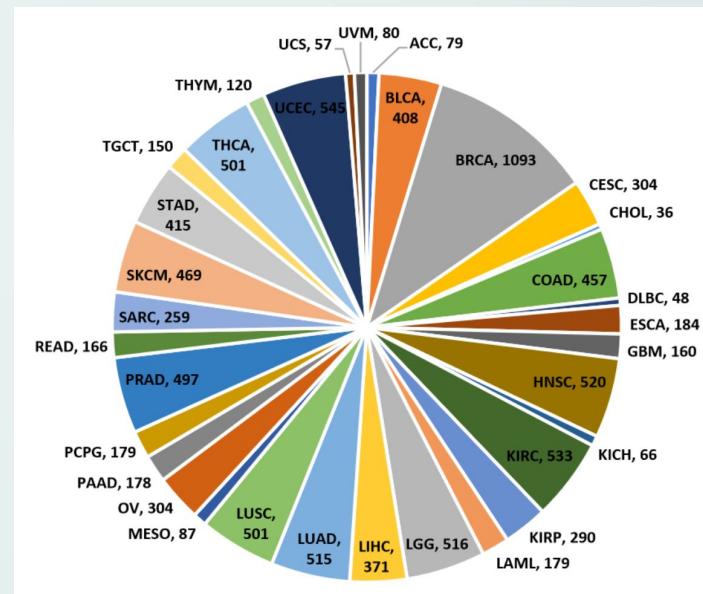
EMBEDDED 2D IMAGES

Trasformazione dei dati in input per il modello

Creazione del dataset

Sono stati utilizzati i **normalized-level3 RNA-Seq gene expression data** di 33 tipi di tumore presenti nel Pan-Cancer Atlas ottenuti dal [Broad GDAC FireHose](#)

- I dati contengono **10.267 campioni di tumore** su 20.531 campioni totali.
- Tali dati sono stati ottenuti tramite l'utilizzo del software **RSEM** che è un pacchetto software di facile utilizzo per **quantificare le abbondanze di geni** e isoforme da dati RNA-Seq single-end o paired-end.
- I **dati**, prima di essere inseriti nel dataset, sono stati **preventivamente ripuliti** dai sample non tumorali ed **etichettati** per consentire l'addestramento della rete neurale.
- Il **dataset finale** ottenuto consiste di un file csv di **circa 2GB** contenente i sample tumorali delle diverse coorti.



Estrazione delle feature (1/3)

Filtering con annotation file

- Il dataset viene dapprima filtrato con l'**homo sapiens annotation file** (aggiornato al 03/04/2018) al fine di eliminare quei geni che non sono presenti in tale file.
- Circa 1000 geni non sono stati trovati nel file di annotation

```
annotation = pd.read_csv(annotation_path, dtype=str)
    # recupero tutti i GeneID dal file di annotation
    gene_id_annotation = list(annotation.loc[:, "GeneID"])
    # recupero tutti i # di cromosoma che servono per l'ordinamento
    gene_id_chr_annotation = list(annotation.loc[:, "chromosome"])
    # La lista seguente conterrà i geneId originali dei raw data
    # (alcuni dei quali non sono presenti nel file di annotation)
    gene_id_original = []
    idx1 = [] # indice dei geni nel raw che trovano una corrispondenza nel file di annotation
    idx1_annotation = [] # indice del geneId del file di annotation dei geni che sono anche in raw data
    k = 0 # indice di progresso
    # nel for si verifica quali geni dei dati raw sono presenti anche nel file di annotation
    for name in feature_name:
        # symbol viene estratta ma non serve in quanto non è mai usata
        symbol, gene_id = name.split("|", 1)
        gene_id_original.append(gene_id)
        if gene_id in gene_id_annotation:
            idx1.append(k)
            # indice del gene nel file di annotation
            idx1_annotation.append(gene_id_annotation.index(gene_id))
    print('compare with annotation, progress : %.2f%%' % ((k / len(feature_name))*100))
    k = k + 1
```

Estrazione delle feature (2/3)

Feature scaling e ordinamento in base al numero cromosomico:

- Tutti i valori vengono dapprima scalati usando **log2 ($y = \log_2(x + 1)$)** così che i valori più **grandi** possano essere **scalati** in un range più **piccolo** al fine di evitare bias
- Poi, tutti i valori **inferiori a 1**, vengono posti a 0 in quanto è altamente probabile che si tratti di rumore
- In seguito, i geni vengono ordinati in maniera naïve in base al numero cromosomico

```
# siccome il range dei valori è enorme è stata applicata una trasformazione ai dati per ridurre la scala
features = np.log2(1.0 + features_raw)
# I valori inferiori a 1 sono stati tutti posti a 0 (zero) in quanto sono ritenuti rumore
features[np.where(features <= 1)] = 0
# [AUTORI] values corresponding to existing genes (1st filtering)
# tutti i geni che hanno trovato una corrispondenza nel file di annotation vengono
# utilizzati come features (dopo la trasformazione con Log2 e l'eliminazione del rumore)
# features[:, idx1_tmp] seleziona solo i geni (le colonne) che sono presenti in idx1
features_filtered = np.array(list(features[:, idx1_tmp] for idx1_tmp in idx1)).transpose()
# recupero i nomi delle feature filtrate
feature_name_filtered = list(feature_name[i] for i in idx1)
# recupero il gene_id delle feature filtrate
gene_id_chr = list(gene_id_chr_annotation[i] for i in idx1_annotation)
# sort the features based on the chr number
idx_sorted = sort_feature(gene_id_chr)
# ordino i nomi delle feature
feature_name_sorted = list(feature_name_filtered[j] for j in idx_sorted)
# features_filtered[:, i] for i in idx_sorted prende i geni in maniera ordinata
features_sorted = np.array(list(features_filtered[:, i] for i in idx_sorted)).transpose()
print('features have been sorted based on chromosome')
```

Estrazione delle feature (3/3)

Filtering basato su Variance Threshold:

- I geni sono filtrati in base alla varianza, in modo da passare da **20531** a **10381** eliminando **i geni** non pertinenti

```
# [GNO] features with a training-set variance Lower than this threshold will be removed
selector = VarianceThreshold(threshold=threshold)
selector.fit(features_sorted)
# maschera di booleani
idx2 = selector.get_support()
# recupero gli indici delle features selezionate (array di interi)
idx2_num = selector.get_support(indices=True)
# [AUTORI] numpy is different from list
features = features_sorted[:, idx2]
feature_name_final = list(feature_name_sorted[i] for i in idx2_num)
feature_name_path = os.path.join(preprocessed_file_path, 'feature_name.csv')
pd.DataFrame(feature_name_final).to_csv(feature_name_path)
print('features are selected, the selected gene name are saved at', feature_name_path)
return features, labels
```

Oversampling

- L'**oversampling** è stato utilizzato al fine di **ovviare** allo **sbilanciamento tra le classi**
- Abbiamo utilizzato l'algoritmo **SVM-SMOTE** che è una **variante di SMOTE** (**Synthetic Minority Oversampling Technique**) che utilizza un algoritmo SVM (Support Vector Machine) per identificare i dati da usare per generare nuovi dati sintetici.
- Per cercare di ottimizzare i tempi, tramite il parametro `sampling_strategy='minority'` abbiamo indicato all'algoritmo di effettuare il resampling solo delle classi minoritarie

```
def over_sampling(training_data, training_label):  
    """  
    This function is used only for classification  
    :param training_data:  
    :param training_label:  
    :return: The training data and training Label after oversampling using SMOTE algorithm  
    """  
    training_data_resampled, training_label_resampled = SVMSMOTE(sampling_strategy='minority', random_state=42, k_neighbors=5, n_jobs=12)\br/>        .fit_resample(training_data, training_label)  
    return training_data_resampled, training_label_resampled
```

Embedding dei dati in immagini 2D

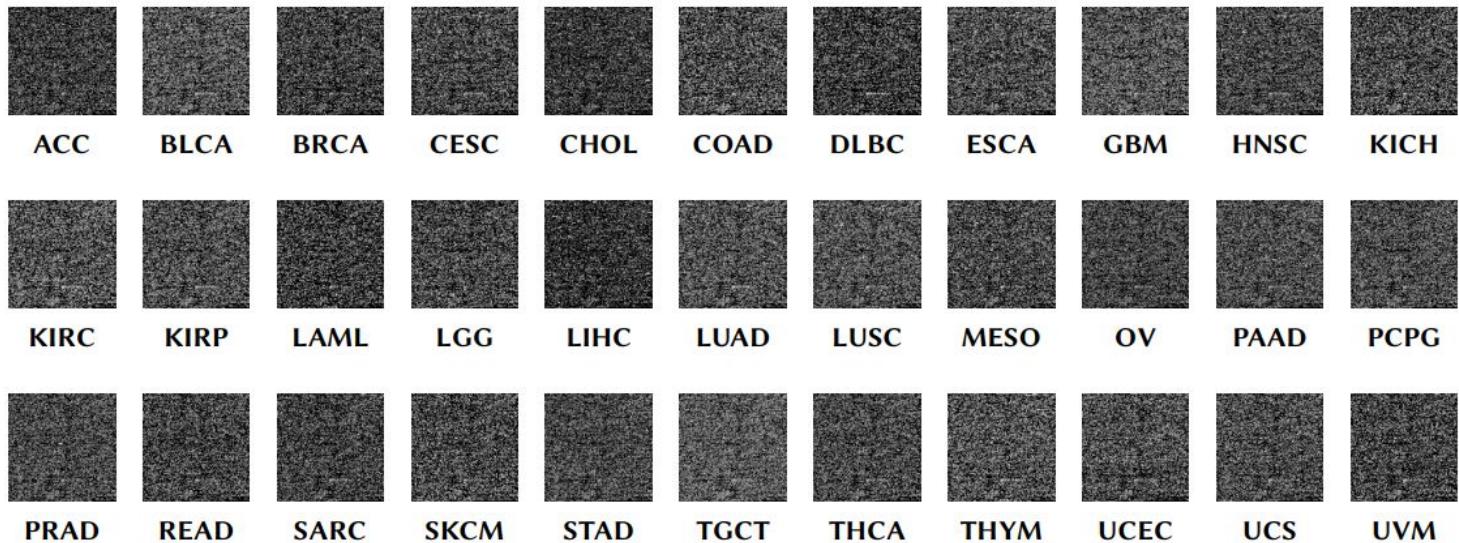
- I dati preprocessati (10381 geni per ogni coorte) vengono inglobati in maniera naïve in immagini 102x102 (10404)
- Gli ultimi valori, di fatti, saranno pixel privi di significato
- Le immagini generate vengono poi divise in fold di training e fold di testing
- Vengono poi salvate, nelle rispettive fold, anche le label associate alle immagini
- A differenza di Lyu e Haque, abbiamo aggiunto la funzione `img_as_ubyte` di scikit-image per evitare un warning sulla lossy conversion delle immagini

```
def embedding_2d(features_train, features_test, fold_idx, preprocessed_file_path, labels_train, labels_test):  
    """  
    This function embed data in 2D images.  
    :param features_train:  
    :param features_test:  
    :param fold_idx: fold index  
    :type fold_idx: int  
    :param preprocessed_file_path:  
    :type preprocessed_file_path: str  
    :param labels_train:  
    :param labels_test:  
    :return:  
    """  
  
    print('embedding to 2D image, fold:', fold_idx)  
    features_padded = np.zeros(img_size*img_size)  
    subfolder_path = os.path.join(preprocessed_file_path + '/img_fold' + str(fold_idx))  
    if not os.path.exists(subfolder_path):  
        os.makedirs(subfolder_path)  
    subfolder_train_path = subfolder_path + '/train'  
    if not os.path.exists(subfolder_train_path):  
        os.makedirs(subfolder_train_path)  
    subfolder_test_path = subfolder_path + '/test'  
    if not os.path.exists(subfolder_test_path):  
        os.makedirs(subfolder_test_path)  
    pd.DataFrame(labels_train).to_csv(os.path.join(subfolder_train_path, 'labels_train.csv'))  
    pd.DataFrame(labels_test).to_csv(os.path.join(subfolder_test_path, 'labels_test.csv'))  
    for i in range(features_train.shape[0]):  
        features_padded[range(features_train.shape[1])] = features_train[i, :]/max(features_train[i, :])  
        features_train_tmp = features_padded.reshape(img_size, img_size)  
        file_path = subfolder_train_path + '/' + str(i) + '.png'  
        # next line is useful for avoiding Lossy conversion warning  
        features_train_tmp = img_as_ubyte(features_train_tmp)  
        io.imwrite(file_path, features_train_tmp)  
    for j in range(features_test.shape[0]):  
        features_padded[range(features_test.shape[1])] = features_test[j, :]/max(features_test[j, :])  
        features_test_tmp = features_padded.reshape(img_size, img_size)  
        file_path = subfolder_test_path + '/' + str(j) + '.png'  
        # next line is useful for avoiding Lossy conversion warning  
        features_test_tmp = img_as_ubyte(features_test_tmp)  
        io.imwrite(file_path, features_test_tmp)
```

Output files

- I file di output sono:
 - Il file **feature_name.csv** che contiene i nomi dei geni estratti nella fase di preprocessing (e che saranno utilizzati in fase di validazione biologica)
 - E le **immagini** necessarie ad eseguire training e testing della rete neurale

```
,0  
0,ABCA4|24  
1,ACOT11|26027  
2,ACOT7|11332  
3,ACTA1|58  
4,ACTL8|81569  
5,ACTN2|88  
6,ADAMTS4|9507  
7,ADAMTS4|54507  
8,ADCY10|55811  
9,ADC|113451  
10,ADORA1|134  
11,ADORA3|140  
12,AFARP1|246182  
13,AGBL4|84871  
14,AGMAT|79814  
15,AGRN|375790  
16,AGT|183  
17,AIM1L|55057  
18,AIM2|9447  
19,AJAP1|55966  
20,AK3L1|205  
21,AK5|26289
```





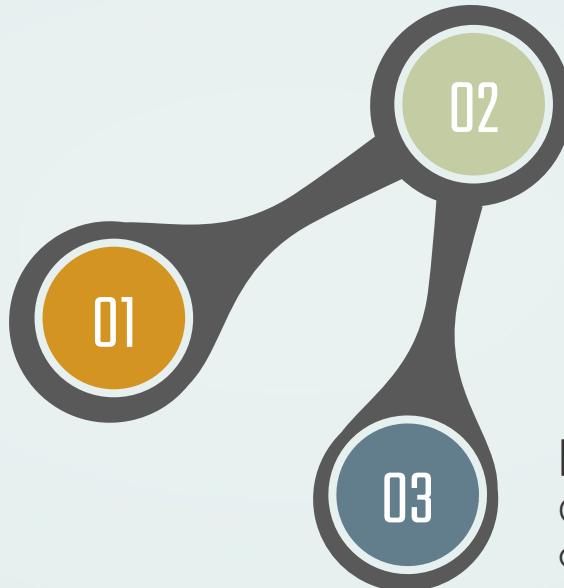
CLASSIFICAZIONE con CNN: Net vs VarNet

Fase di addestramento e test delle
reti neurali

HIGHLIGHTS

MANIPOLAZIONE DEL DATASET

Controllare il dataset per darlo in pasto al modello



VARNET

Cambiare le CNN usando principi di reti pretrained

NET VS VARNET

Confronto sulla classificazione



Data Augmentation (1/2)



- I risultati della **pre-elaborazione** dei dati così come sono non possono essere compresi dal modello, per cui va effettuata una trasformazione dei dati in **tensori** questi risultano essere un formato comprensibile per la rete.
- La Data Augmentation ci viene in nostro aiuto essa viene definita quando usata nel Deep Learning come:

“Data augmentation is the process of transforming images to create new ones, for training deep learning models.”

- Il principale **beneficio** è che **trasformando i dati** in una forma comprensibile, si riduce il rischio di **overfitting** del modello.
- Alcuni esempi sono:
 - **Flipping**
 - **Rotation**
 - **GridMask**



Data Augmentation (2/2)



```
class ToTensor(object):

    def __call__(self, sample):
        image, labels = sample['image'], sample['label']
        return {'image': torch.from_numpy(image), 'label': torch.LongTensor(
            labels)} # torch.LongTensor rappresenta il cambio di tipo da tensore 1x102x102 in un intero a 64 bit, variante utilizzata per le GPU
```

```
test_dataset = supcls.LocalTumorDatasetTest(csv_file=test_csv_path,
                                              root_dir=root_test_path + str(fold_idx) + '/test',
                                              transform=transforms.Compose([supcls.ToTensor()]))
```

- Nel nostro lavoro il compito descritto in precedenza, viene svolto dalla classe **ToTensor**, che esegue i seguenti passi:
 - Prendendo in **input** un'immagine del campione, prima separa le immagini dalla sua etichetta;
 - Converte le **immagine** e **l'etichetta** nei rispettivi tensori.
 - Restituisce un **dictionary** che contiene i nuovi valori **tensoriali**.
- Quanto appena detto è vero per la singola immagine, per fare ciò sull'intero **dataset**, richiamiamo la funzione **Compose**, che viene passata al dataset tramite il parametro **transform**.

Dataset e DataLoader (1/2)

```
class LocalTumorDatasetTrain(Dataset):
    def __init__(self, csv_file, root_dir, transform=None):
        self.labels_frame = np.array(pd.read_csv(csv_file, skiprows=1, sep=',', header=None)).astype(int)
        self.root_dir = root_dir
        self.transform = transform

    def __len__(self):
        return len(self.labels_frame)

    def __getitem__(self, idx):
        img_name = str(idx) + '.png'
        img_path = os.path.join(self.root_dir, img_name)
        img = np.empty(shape=(1, 102, 102))
        img[0, :, :] = (img_as_float(
            io.imread(img_path)) - 0.5) / 0.5      # riduce il numero di cifre a 4 per ogni valore del tensore dell'immagine
        label = np.array([self.labels_frame[idx, 1]]) # estrae il valore tensoriale della label
        train_sample = {'image': img, 'label': label}
        if self.transform:
            train_sample = self.transform(train_sample)
        return train_sample
```

```
dataloader_train = supcls.DataLoader(train_dataset, batch_size=batch_size, shuffle=True, num_workers=0)
dataloader_test = supcls.DataLoader(test_dataset, batch_size=batch_size, shuffle=True, num_workers=0)
```

- Qui si mostra la **classe** che il compito di eseguire tutti i metodi necessari alla **gestione** del dataset:
 - **Inizializzare**
 - **Stabilire la lunghezza del dataset**
 - **Recuperare un campione e restituirlo**
- In seguito necessitiamo di una classe che conservi ed faccia iterazione sul dataset appena generato, qui ci viene in aiuto **DataLoader**.

Dataset e DataLoader (2/2)

```
class LocalTumorDatasetTest(Dataset):

    def __init__(self, csv_file, root_dir, transform=None):
        self.labels_frame = np.array(pd.read_csv(csv_file, skiprows=1, sep=',', header=None)).astype(int)
        self.root_dir = root_dir
        self.transform = transform

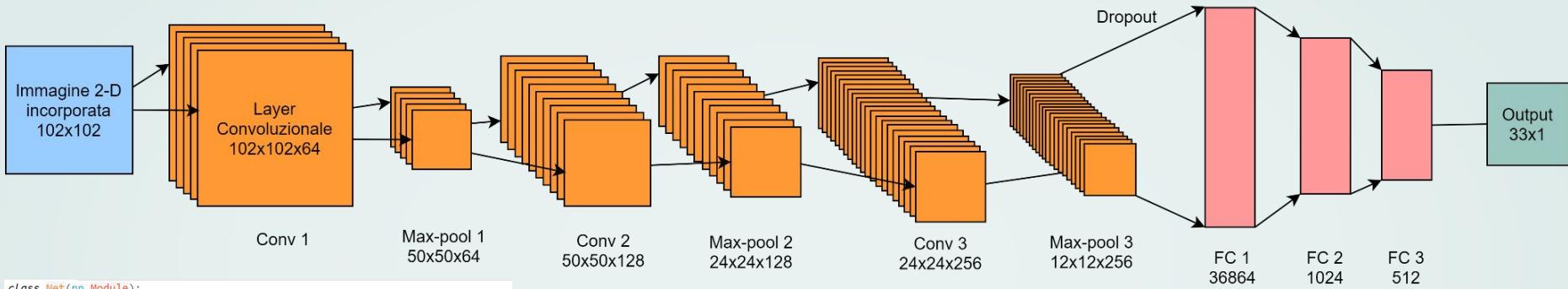
    def __len__(self):
        return len(self.labels_frame)

    def __getitem__(self, idx):
        img_name = str(idx) + '.png'
        img_path = os.path.join(self.root_dir, img_name)

        """ Setting dimensione immagine !!!"""
        img = np.empty(shape=(1, 102, 102))
        img[0, :, :] = (img_as_float(io.imread(img_path)) - 0.5) / 0.5
        label = np.array([self.labels_frame[
            idx, 1]])
        test_sample = {'image': img, 'label': label}

        if self.transform:
            test_sample = self.transform(test_sample)
        return test_sample
```

Architettura di Net



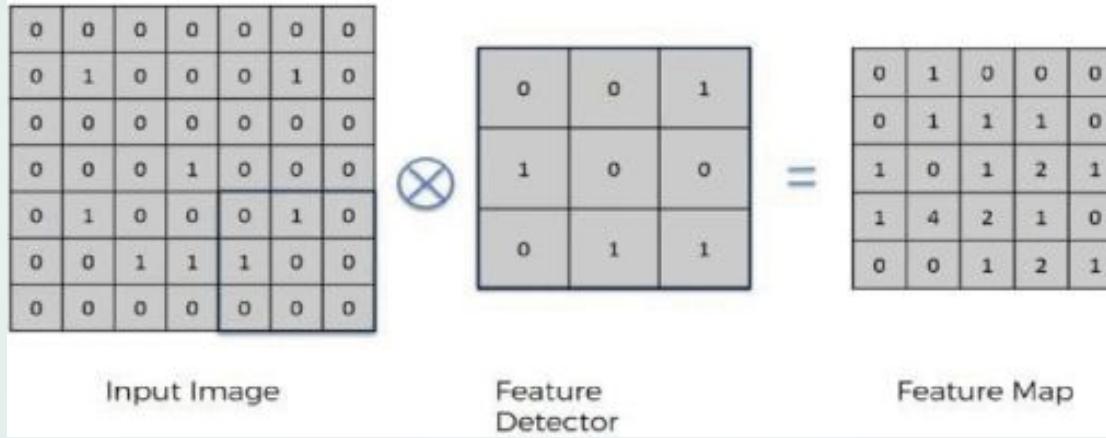
```
class Net(nn.Module):
    def __init__(self, num_of_classes):
        super(Net, self).__init__()
        # input image channel, output channels, kernel size square convolution
        # kernel
        # input size = 102, output size = 100
        self.conv1 = nn.Conv2d(1, 64, kernel_size=5, padding=1)
        self.bn1 = nn.BatchNorm2d(64)
        # input size = 50, output size = 48
        self.conv2 = nn.Conv2d(64, 128, kernel_size=5, padding=1)
        self.bn2 = nn.BatchNorm2d(128)
        # input size = 24, output size = 24
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(256)
        self.drop2D = nn.Dropout2d(p=0.25, inplace=False)
        self.vp = nn.MaxPool2d(kernel_size=2, padding=0, stride=2)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(256 * 12 * 12, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, num_of_classes)

    def forward(self, x):
        in_size = x.size(0)
        x = f.relu(self.bn1(self.vp(self.conv1(x))))
        x = f.relu(self.bn2(self.vp(self.conv2(x))))
        x = f.relu(self.bn3(self.vp(self.conv3(x))))
        x = self.drop2D(x)
        x = x.view(in_size, -1)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        return x
```

- Lo schema della CNN usata in questo lavoro è così strutturato:
 - **3 layer Convoluzionali**
 - **3 layer di Max-Pooling**
 - **3 layer di Batch Normalization**
 - **1 layer di Drop-out**
 - **3 layer Fully Connected**
- Per un totale di 13 strati nascosti di cui solo 9 di essi possiedono dei parametri addestrabili.

Hidden Layers di Net (1/6)

Convolutional Layer



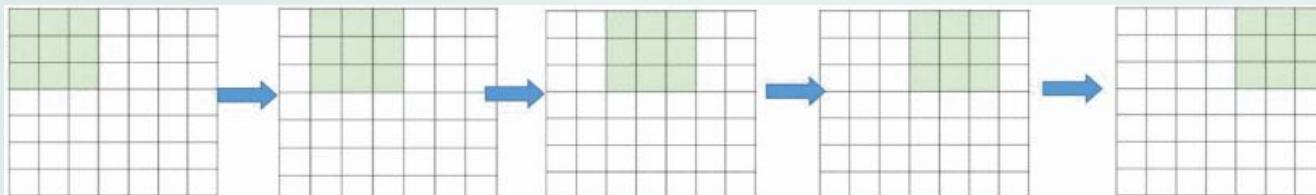
- Questo è il layer più importante della rete, esegue l'operazione di convoluzione dell'immagine d'input generando quindi una feature map, che memorizza le caratteristiche riducendo la dimensione dell'immagine in uscita (**downsampling**).

Hidden Layers di Net (2/6)

Convolutional layer: padding e stride

0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

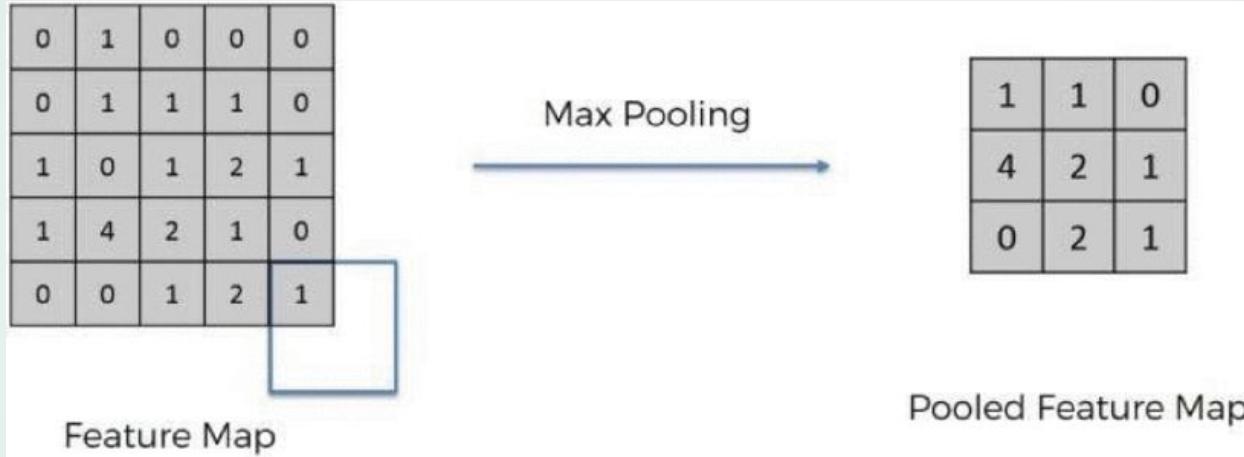
- **Stride** e **padding** sono due ulteriori parametri che migliorano l'efficienza della **convoluzione**.
- Il primo risolve il problema della **sovraposizione** dei vari filtri applicati che osservano le regioni dell'immagine.
- Il secondo invece risolve la problematica della **perdita d'informazione** sui pixel del bordo, oltre ciò ci permette di gestire la **dimensione** dell'output.



$$O = 1 + \frac{N + 2P - F}{S}$$

Hidden Layers di Net (3/6)

Max pooling layer



Il **pooling** è un passo importante per **ridurre** ulteriormente le dimensioni della mappa di attivazione, mantenendo solo le caratteristiche importanti e riducendo **l'invarianza spaziale**. Questo a sua volta riduce il numero di caratteristiche **apprendibili** dal modello. Ciò aiuta a risolvere il problema del sovradimensionamento. Nel nostro **lavoro** viene usato il **Max Pooling** che prende il **valore più alto** da ogni sub matrice della mappa di attivazione e forma una matrice separata. In questo modo si assicura che le **caratteristiche apprendibili** rimangano limitate nel numero, preservando allo stesso tempo le caratteristiche chiave di qualsiasi immagine.

Hidden Layers di Net (4/6)

Batch normalization layer

- **BatchNorm** è un meccanismo che mira a **stabilizzare la distribuzione** (su un mini lotto) di input ad un determinato livello di rete durante **l'addestramento**. Ciò si ottiene aumentando la rete con strati aggiuntivi che impostano i primi **due momenti** (media e varianza) della distribuzione di ciascuna attivazione rispettivamente a 0 e 1.
- Il **fulcro** del successo di BatchNorm è data dalla risoluzione al problema della **Internal Covariate Shift** definito come:

""We define Internal Covariate Shift as the change in the distribution of network activations due to the change in network parameters during training..

- Questo cambiamento porta ad un **costante spostamento** del problema di formazione di base e si ritiene quindi che abbia un **effetto negativo** sul processo di training.

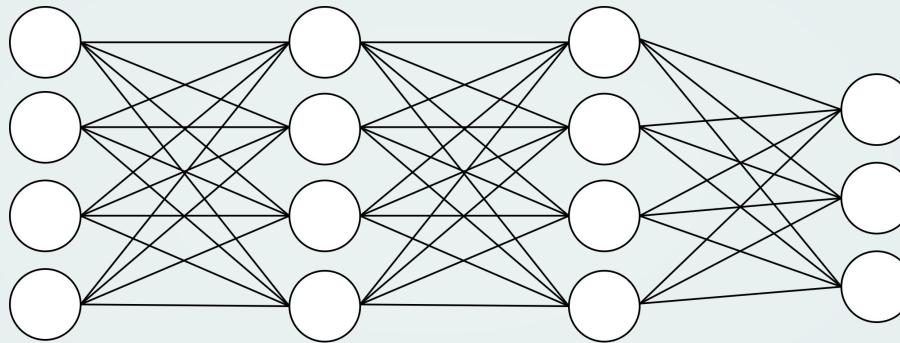
Hidden Layers di Net (5/6)

Drop-out layer

- Il **drop-out** è un metodo di **regolarizzazione** che imposta stocasticamente a zero le attivazioni delle unità nascoste per ogni caso di addestramento al momento della fase di training. Questo interrompe il **co-adattamento** dei rilevatori di funzionalità poiché le unità degli strati nascosti selezionati dal drop-out non possono influenzare le restanti unità.
- È simile al **bagging**, in cui un insieme di modelli sono addestrati su diversi sottoinsiemi degli stessi dati di allenamento. Al momento della prova, le previsioni dei diversi modelli sono mediate insieme. Nel bagging, ogni modello ha parametri indipendenti e tutti i membri sarebbero addestrati individualmente. Nel caso di formazione tramite drop-out, ci sono esponenzialmente molti modelli eventualmente da addestrare ma non lo sono tutti in maniera palese, poiché questi ultimi condividono gli stessi parametri.
- In realtà, il numero di modelli addestrati tramite drop-out non è più grande di $\gamma \times \epsilon$, dove γ è il numero di esempi di formazione, ed ϵ è il numero di epoche usate nel training.

Hidden Layers di Net (6/6)

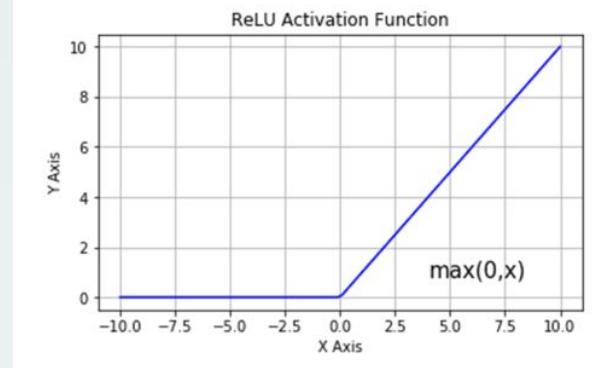
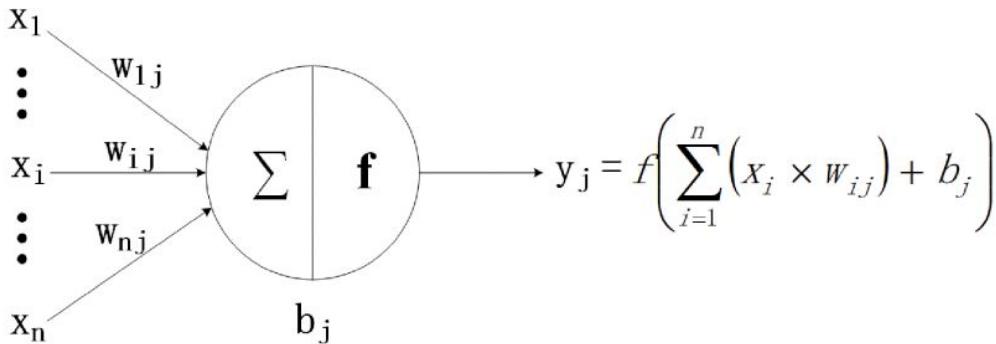
Layer fully-connected



Layer Fully Connected

Questo **layer** viene utilizzato alla fine della rete neurale. Generalmente la **matrice dei pesi** viene appiattita prima di essere passata ai neuroni. È difficile seguire **i dati** dopo questo punto a causa della presenza di molte **unità nascoste** con peso variabile all'uscita di ogni neurone. Tutto il **ragionamento** e il **calcolo** sui dati ed il raggiungimento di una **target class** associato al nostro input, viene delegato a questa tipologia di layer.

Funzioni di attivazione



- Le **CNN** possono sfruttare diverse **funzioni di attivazione** per esprimere **funzioni complesse**. Simile alla **funzione** del modello del **neurone** del cervello umano, la funzione di attivazione qui è un'unità che determina quale **informazione** dovrebbe essere trasmessa **al neurone seguente**. Ogni neurone nella rete neurale accetta il valore di uscita dei neuroni dal livello precedente come input e passa il valore elaborato al livello successivo. In una **rete neurale multilayer**, la funzione di attivazione è posta tra due layer.
- Per il nostro lavoro viene usata la **ReLU** che risulta essere una funzione di attivazione efficace. Quando x è inferiore a 0, il suo valore di funzione è 0; quando x è maggiore o uguale a 0, il suo valore di funzione è x stesso.

Funzioni di perdita

- **La funzione di perdita** o la funzione di costo è sfruttata per calcolare la distanza fra il valore previsto ed il valore reale. La funzione di perdita è usata solitamente come criterio imparante del problema di **ottimizzazione**. La funzione di perdita può essere utilizzata con CNN per affrontare i problemi di regressione e i problemi di classificazione, il cui obiettivo è quello di minimizzare la funzione stessa.
- Per il nostro lavoro è stata usata la **Cross-Entropy**, che è usata per valutare la differenza tra la distribuzione di probabilità ottenuta dall'allenamento corrente e la distribuzione effettiva. Questa funzione confronta la probabilità prevista con il valore di uscita effettivo (0 o 1) in ogni classe e calcola il valore di penalità in base alla distanza da loro. La penalità è logaritmica, pertanto, la funzione fornisce una penalità minore (0.1 o 0.2) per differenze più piccole tra le due distribuzioni e una penalità più grande (0.9 o 1.0) per differenze più grandi.

Ottimizzatore

- Nelle CNN, spesso abbiamo bisogno di ottimizzare le funzioni non convesse. I metodi matematici richiedono un'enorme potenza di calcolo, pertanto, gli ottimizzatori vengono utilizzati nel processo di allenamento per ridurre al minimo la funzione di perdita e per ottenere parametri di rete ottimali in un tempo accettabile.
- **L'ottimizzatore** utilizzato nel nostro lavoro è la stima adattiva del momento (**Adam**). È essenzialmente un algoritmo formato combinando la quantità di moto con **I'RMSprop**. Adam memorizza sia la media di decadimento esponenziale dei gradienti quadrati passati, come l'algoritmo Adadelta, sia la media di decadimento esponenziale medio dei gradienti passati, come l'algoritmo di quantità di moto.

Configurazione Iperparametri

- Quando si costruisce una CNN, oltre a selezionare la funzione di attivazione, la funzione di perdita e l'ottimizzatore, dobbiamo anche regolare molti altri iperparametri che influiscono notevolmente sulle prestazioni del modello.
- Come è noto, non esiste un insieme fisso di iperparametri in grado di garantire una soluzione ottimale per tutto il tempo.
- Per cui, un insieme di esperienze e buone regole pratiche sono significative durante l'hyperparameter tuning.
- Un modello di deep neural network (DNN) apprende i valori appropriati delle sue variabili di configurazione, cioè i pesi di connessione e il bias, dai dati di allenamento e, tali variabili, sono chiamate iperparametri.
- Nel nostro lavoro sono configurati i seguenti iperparametri legati alla fase di training:
learning rates, number of epoch e batch_size.

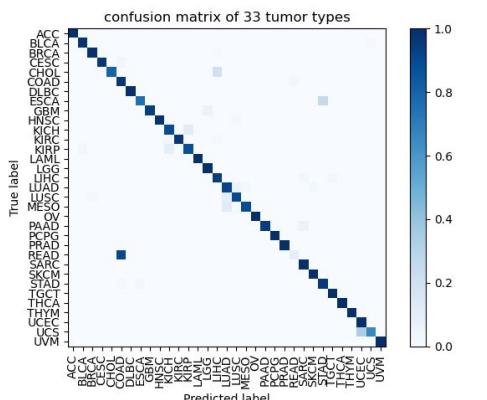
Risultati Training Net (GPU)

Fold ID	Accuracy	Epoche	Tempo
0	94%	44	2:40:24
1	95%	131	8:00:44
2	95%	49	2:56:24
3	95%	50	3:00:12
4	95%	200	12:10:15
5	94%	200	12:00:00
6	94%	43	2:34:48
7	95%	49	2:57:09
8	95%	39	2:21:24
9	96%	51	2:30:36

Risultati Classificazione Net (CPU vs GPU)

Matrice di confusione, evaluation metrics

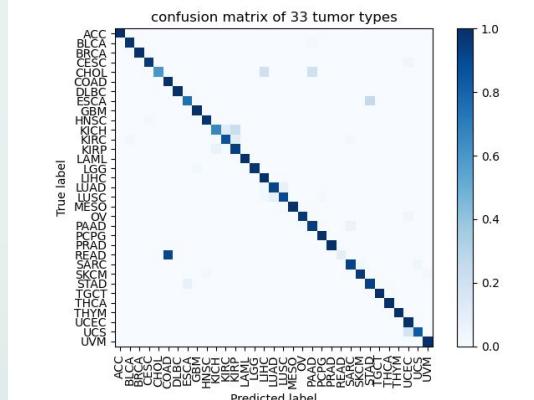
Metodo	Accuracy	Precision (P)	Recall (R)	F1-score (F1)
CNN-CPU	95.79%	95.95%	95.79%	95.57%
CNN-GPU	94.74%	95.10%	94.74%	94.37%



CPU

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = \frac{2PR}{P + R}$$



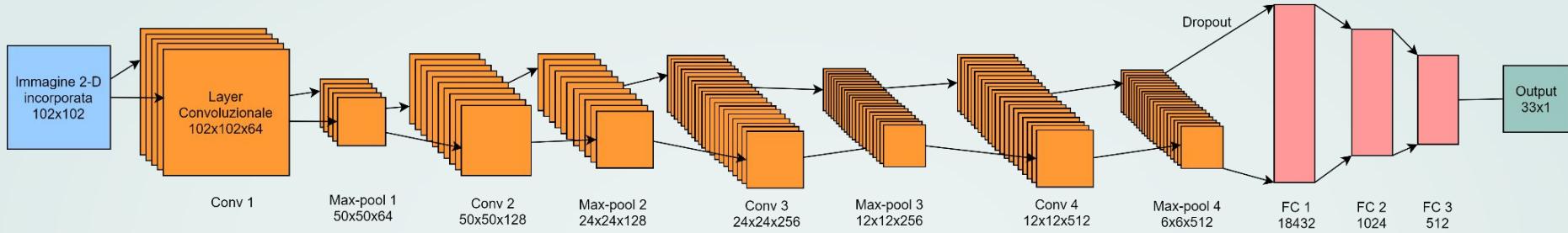
GPU

VarNet

- Analizzando la metodologia applicata alla rete CNN di riferimento (per ulteriori dettagli consultare la sezione 3.4) ed i risultati ottenuti (cfr. 4) sulla pipeline dell'intero progetto ci siamo posti la seguente domanda:

“Quali sono i punti salienti che possono rendere l'individuazione dei biomarker più efficiente, affidabile e veloce?”
- Dalla precedente domanda nasce VarNet che sta per: Variant Convolutional Neural Network, come sforzo di migliorare Net sulle performance e sui tempi di training.
- Le modifiche che hanno portato a VarNet partendo da Net sono essenzialmente 2:
 - a. **Architettura**
 - b. **Learning blocks**

Architettura di VarNet



```

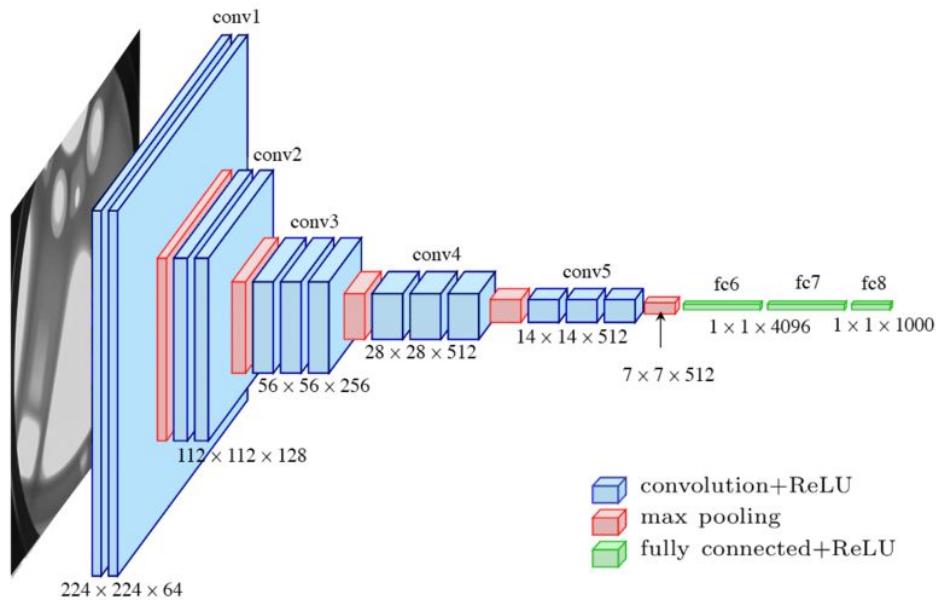
class VarNet(nn.Module):
    def __init__(self, num_of_classes):
        super(VarNet, self).__init__()
        # input image channel, output channels, kernel size square convolution
        # kernel
        # input size = 102, output size = 100
        self.conv1 = nn.Conv2d(1, 64, kernel_size=5, padding=1)
        self.bn1 = nn.BatchNorm2d(64)
        # input size = 50, output size = 48
        self.conv2 = nn.Conv2d(64, 128, kernel_size=5, padding=1)
        self.bn2 = nn.BatchNorm2d(128)
        # input size = 24, output size = 24
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(256)
        # input size = 12, output size = 12
        self.conv4 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(512)
        self.drop2D = nn.Dropout2d(p=0.25, inplace=False)
        self.vp = nn.MaxPool2d(kernel_size=2, padding=0, stride=2)
        # an affine operation: y = w*x + b
        self.fc1 = nn.Linear(512 * 6 * 6, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, num_of_classes)

    def forward(self, x):
        in_size = x.size(0)
        x = F.relu(self.bn1(self.vp(self.conv1(x))))
        x = F.relu(self.bn2(self.vp(self.conv2(x))))
        x = F.relu(self.bn3(self.vp(self.conv3(x))))
        x = F.relu(self.bn4(self.vp(self.conv4(x)))) # Selu must be explore because overcame the problem of negative slopes that afflict relu
        x = self.drop2D(x)
        x = x.view(in_size, -1)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        return x

```

- Lo schema di VarNet strutturato in questo lavoro è composto da:
 - **4 layer Convoluzionali**
 - **4 layer di Max-Pooling**
 - **4 layer di Batch Normalization**
 - **1 layer di Drop-out**
 - **3 layer Fully Connected**
- Per un totale di 16 strati nascosti di cui solo 11 di essi possiedono dei parametri addestrabili.

Differenze con Net



Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).

- **Architettura:** In VarNet, dunque, viene aggiunto prima del layer di drop-out un quarto layer convoluzionale "conv4" contenente 512 filtri, il suddetto valore è stato scelto seguendo in parte il principio di VGG e posti immediatamente dopo di esso, un layer di Max-Pooling e Batch Normalization.
 - Oltre ciò cambiano le dimensioni del primo layer fully-connected che risulta essere di 18432.
- **Learning blocks:** li definiamo come i componenti fondamentali che modellano tutti gli aspetti dell'apprendimento necessari all'efficacia della rete feedforward. Essi sono: Activation Function, Loss Function e Optimizer.
 - Activation Function: SELU vs ReLU
 - Loss Function: Margin Loss vs Cross Entropy
 - Optimizer: Nadam vs Adam

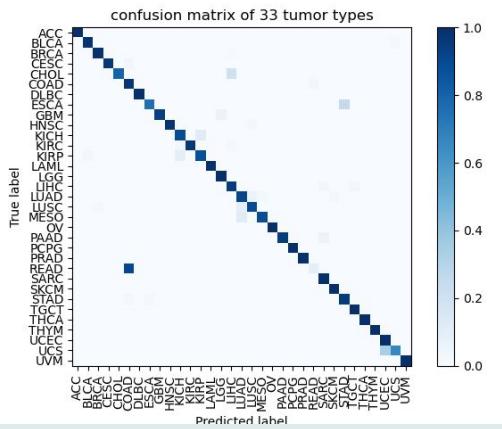
Risultati Training VarNet (GPU)

Table 12. Tabella del training GPU su dataset oversampled di VarNet

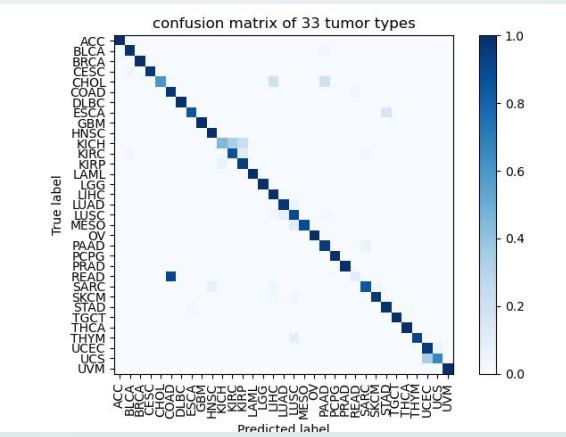
Fold ID	Accuracy	Epoche	Tempo
0	94%	200	14:00:34
1	95%	200	14:05:03
2	95%	189	13:22:22
3	95%	200	14:06:19
4	95%	200	14:06:19
5	95%	200	14:06:44
6	93 %	200	14:06:45
7	95%	200	14:07:14
8	95%	200	14:07:05
9	96%	200	14:07:50

Risultati Classificazione VarNet (CPU vs GPU)

Metodo	Accuracy	Precision	Recall	F1-score
CNN-CPU	95.41%	95.39%	95.41%	95.12%
CNN-GPU	94.83%	94.66%	94.83%	94.45%



CPU



GPU



Output files



- Gli **output files** di questo modulo sono composti dai file generati dalla libreria **PyTorch**:
 - network_weights.pth (pesi della rete Net)
 - network_weights_variant.pth (pesi della rete VarNet)

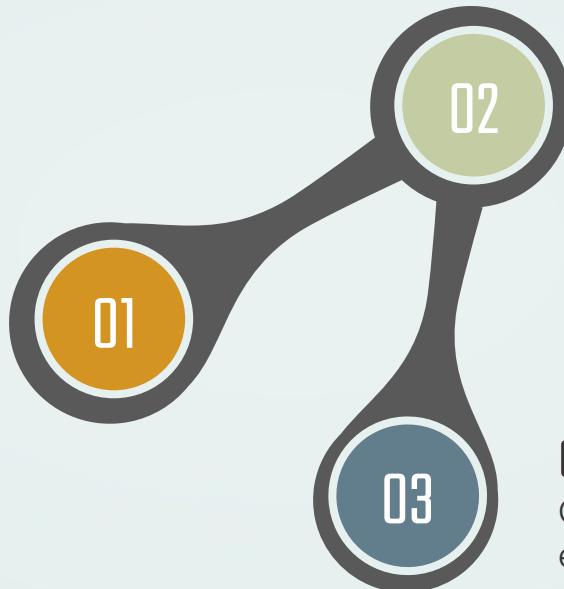
INTERPRETABILITÀ: Guided Grad-CAM

Comprendere le decisioni delle
DNN



HIGHLIGHTS

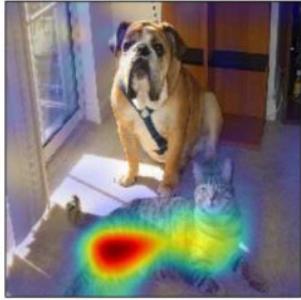
Grad-CAM
Capire l'importanza di ogni neurone per generare la mappa di localizzazione



Guided Backpropagation

Guidare la propagazione tramite dei vincoli sull'input e sull'errore

Guided Grad-CAM
Combinazione di Grad-CAM e Guided Backpropagation



(c) Grad-CAM 'Cat'



(i) Grad-CAM 'Dog'

- Questo peso α_k^c rappresenta una linearizzazione parziale della rete profonda a valle di A , e cattura l'importanza della mappa caratteristica k per una classe target c . Eseguiamo una combinazione ponderata di mappe di attivazione in avanti e applichiamo ReLU per ottenere:

Gradient Classification Activation Map

- Grad-CAM usa le informazioni di gradiente che fluiscono nell'ultimo strato convoluzionale della CNN per capire l'importanza di ogni neurone per una decisione di interesse.
- al fine di ottenere la mappa di localizzazione classe-discriminante Grad-CAM del layer L definita come $L_{\text{Grad-CAM}}^c \in \mathbb{R}^{u \times v}$ di larghezza u e altezza v per qualsiasi classe c , in primo luogo calcoliamo il gradiente del punteggio per la classe c , y^c (prima del Softmax), rispetto alla mappa delle caratteristiche A^k di uno strato convoluzionale, cioè $\partial y^c / \partial A^k$.
- Questi gradienti che scorrono indietro sono raggruppati nella media globale per ottenere i pesi α_k^c dell'importanza dei neuroni

$$\alpha_k^c = \underbrace{\frac{1}{Z} \sum_i \sum_j}_{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left(\underbrace{\sum_k \alpha_k^c A^k}_{\text{linear combination}} \right)$$



(b) Guided Backprop 'Cat'



(h) Guided Backprop 'Dog'

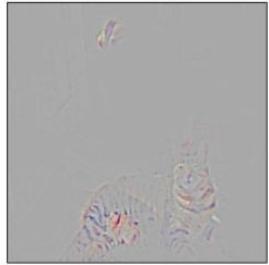
Guided Backpropagation

- Parte del potere d'interpretazione visiva di Guided Grad-CAM è ottenuta utilizzando l'algoritmo di **Guided Backpropagation** basato sul gradiente (GBP) che si concentra sui pixel rilevanti delle immagini, responsabili della **previsione della tipologia tumorale**, fornendo un **punteggio di confidenza** a ciascun pixel.
- L'algoritmo GBP esegue una propagazione in avanti attraverso il modello in cui l'input viene passato attraverso la funzione di attivazione ReLU (z_i^l rappresenta il punteggio di attribuzione per ogni timestamp)
- Durante la propagazione all'indietro, le saliency map ottenute con il filtro di convoluzione **passano solo i valori** che sono stati **positivi** durante il passaggio in avanti e **tagliano il resto dei valori**.
- Il GBP modifica la propagazione all'indietro classica, aggiungendo una condizione: passano solo i valori che sono stati positivi durante la propagazione in avanti e all'indietro.
- Dato questo vincolo viene generato un nuovo segnale di errore che tiene conto sia di z_i^l che di E_i^{l+1} prima di passare l'errore al livello precedente.
- il gradiente è quindi guidato dall'input dello strato precedente e dal segnale di errore dello strato successivo.

$$z_i^{l+1} = \text{ReLU}(z_i^l) = \max(z_i^l, 0)$$

$$E_i^l = E_i^{l+1} \quad \forall (z_i^l > 0), \text{ where } E_i^{l+1} = \frac{\delta z^{out}}{\delta z_i^{l+1}}$$

$$E_i^l = E_i^{l+1} \quad \forall (z_i^l > 0) \text{ and } (E_i^{l+1} > 0)$$



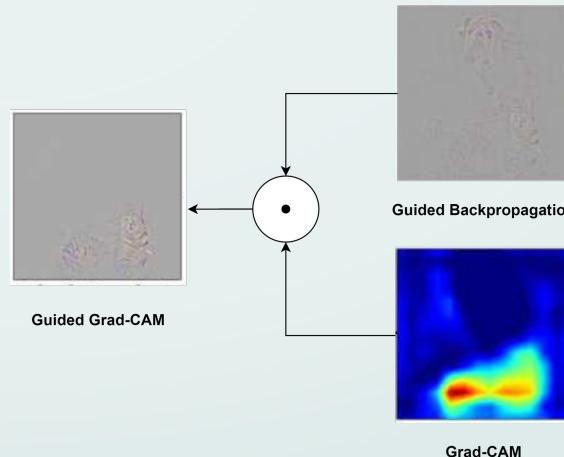
(d) Guided Grad-CAM ‘Cat’



(j) Guided Grad-CAM ‘Dog’

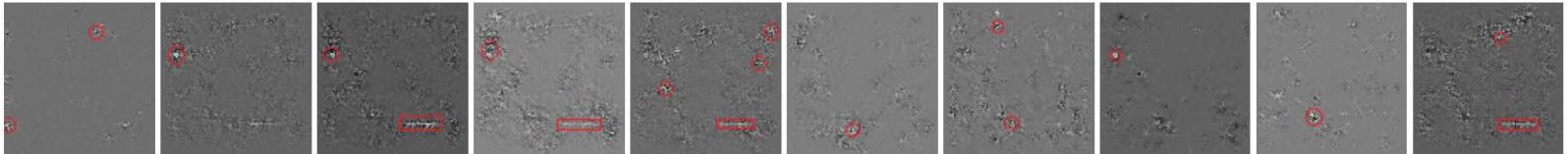
Guided Grad-CAM

- Mentre le visualizzazioni Grad-CAM sono class-discriminat e localizzano bene le regioni più rilevanti in un’immagine, esse non hanno la capacità di mostrare l’importanza a grana fine come i metodi di visualizzazione gradient pixel-space transformation
- Quindi, al fine di combinare gli aspetti migliori di entrambe le tecniche, esse vengono fuse tramite moltiplicazione puntiforme.
- Le heatmap generate da questa fusione risultato essere class-discriminative e ad alta risoluzione



Guided Grad-CAM di classe tumorale

DLBC



- Per generare le heatmap di ogni coorte tumorale, dapprima viene fatta una media e poi i valori vengono normalizzati nel range [0, 255]
- Tale passo, da noi inserito, non era stato previsto dagli autori del paper di riferimento.

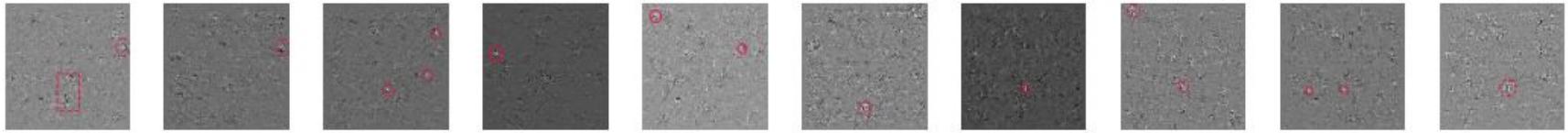


Output files

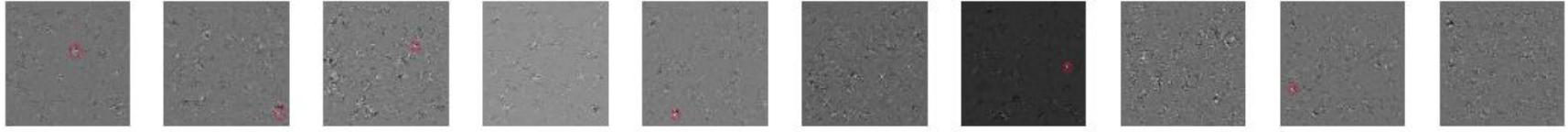


- Gli output files di questo modulo sono le heatmap generate per ogni coorte tumorale

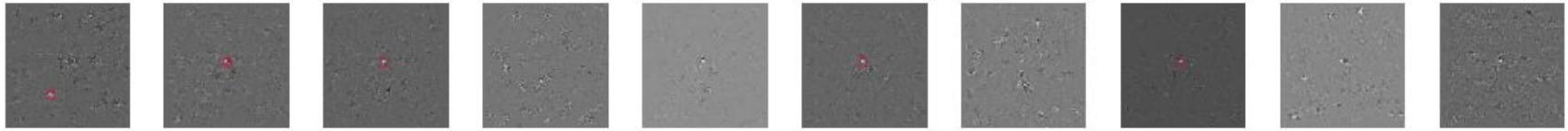
BLCA



LGG



PRAD



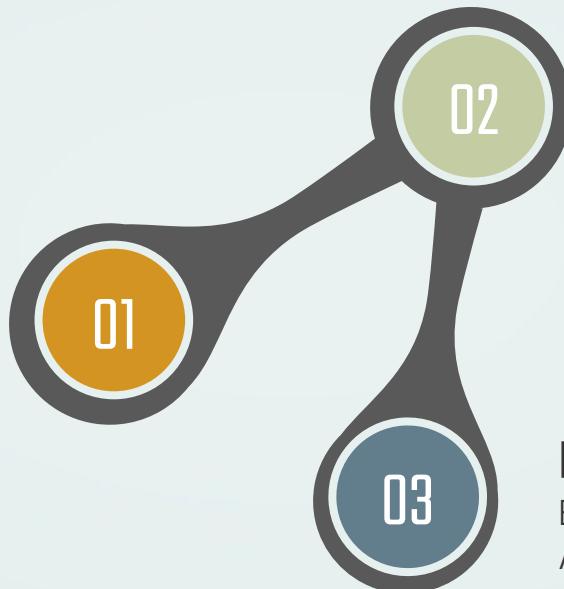
VALIDAZIONE dei BIOMARKER

Convalidare i risultati biologici
ottenuti



HIGHLIGHTS

**ESTRAZIONE dei
CONFIDENCE SCORE**
Comprendere l'importanza
dei geni



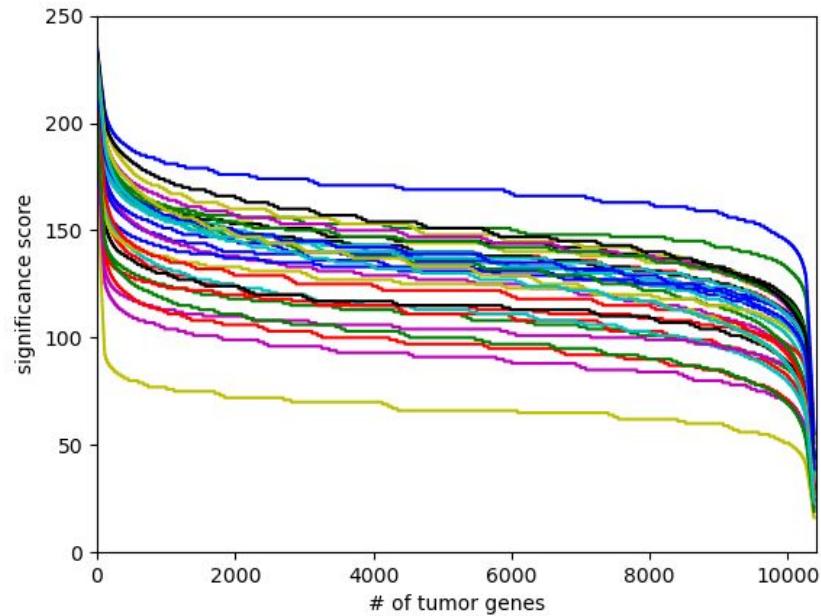
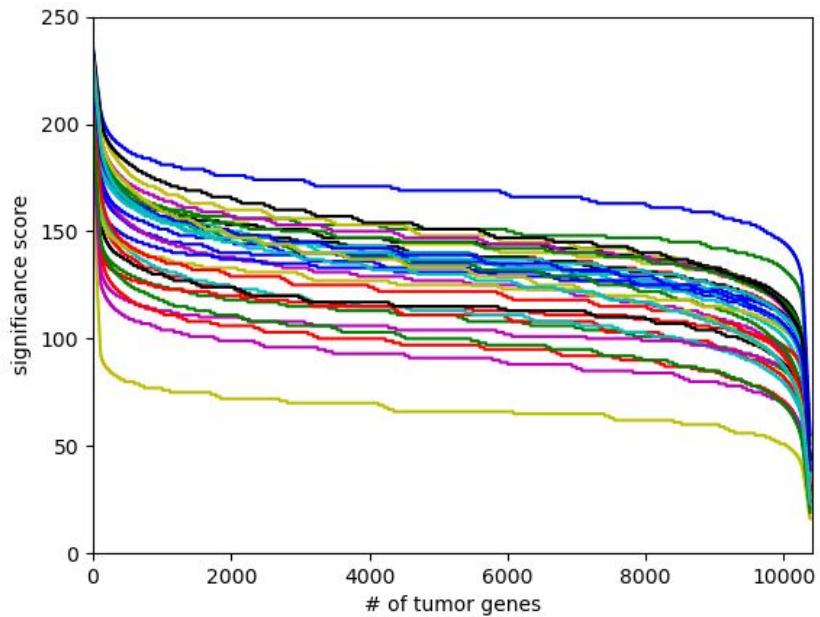
**PLOT dei DATI e
ESTRAZIONE DELLE LISTE di GENI**
Visualizzare il grafico del
confidence score e estrarre
le liste di geni per la
pathway analysis

PATHWAY ANALYSIS
Effettuare la Pathway
Analysis con il tool DAVID

Estrazione dei Confidence Score

```
def gene_confidence_score_heatmaps(out_path_base, classes):
    """
    Function to get and save the confidence score from Guided Grad-CAM images
    :param out_path_base: base path to save data
    :type out_path_base: str
    :param classes: list of classes names
    :type classes: list
    :return: None
    """
    for folder_no in range(0, support.NUM_FOLDERS):
        out_path = out_path_base + str(folder_no)
        for i in tqdm(range(0, Len(classes))):
            # CAUTION ! Make sure to set the correct path for heatmaps in function transform_img_to_data in support.py
            D = support.transform_img_to_data(out_path_base, folder_no, classes[i])
            if not os.path.exists(os.path.join(out_path, 'ConfidenceScore')):
                os.makedirs(os.path.join(out_path, 'ConfidenceScore'))
            D.to_csv(os.path.join(out_path, 'ConfidenceScore', classes[i] + '_confidence.csv'), header=False, index=False)
        print('Complete to save the trasform img to data for folder:', folder_no)
```

Plot: confidence score vs rank genes



PFKM

MVP

COX6B2

PEBP1

LOC642597

C12orf34

CACNA1H

MIR17HG

FAM19A4

CYP17A1

TUBB4

TMEM132A

PHLDA1

DLK1

SYNGR2

ATF5

MT3

VANIKO

Top 400 genes

```
# dictionary_rank_genes-confidence_score
    for num in range(0, len(dictionaries)):
        df = pd.DataFrame(dictionaries[num], index=[0])
        if not os.path.exists(img_dir_base + str(folder_no) + '/top_genes/'):
            os.makedirs(img_dir_base + str(folder_no) + '/top_genes/')
        df.to_csv(img_dir_base + str(folder_no) + '/top_genes/' + classes[
            num] + '_dictionary_rank_genes-confidence_score.csv', header=False, index=False)

    classes_filtered_lists = []
    for cls in range(0, len(classes)):
        filter_list = []
        counter_filter = 0
        for key in dictionaries[cls]:
            filter_list.append(key)
            counter_filter += 1
            if counter_filter == GENE_FILTER_SIZE:
                break
        classes_filtered_lists.append(filter_list)

# saving top 400 genes list in a .txt file for submission
path_file = img_dir_base + str(folder_no)
for cls in range(0, len(classes)):
    filename = path_file + '/top_genes/' + classes[cls] + '_top400thgenes_fold' + str(folder_no) + '.txt'
    if os.path.exists(filename):
        os.remove(filename)

    file = open(filename, 'w')
    for gene_name in classes_filtered_lists[cls]:
        file.write(gene_name + "\n")
    file.close()

print("Complete save files for DAVID Platform for folder_no: ", folder_no)
```

DAVID platform

Database for Annotation, Visualization and Integrated Discovery

The screenshot shows the DAVID platform's main interface. At the top, there's a navigation bar with links for Home, Start Analysis, Shortcut to DAVID Tools, Technical Center, Downloads & APIs, Term of Service, About DAVID, and About LHRI. On the left, there's a sidebar titled 'Gene List Manager' with options like 'Upload', 'List', 'Background', 'Select Species', 'List Manager', 'Help', and 'DLBC_geneToSubmit'. The main content area is titled 'Annotation Summary Results' and displays the following information:

- Current Gene List: DLBC_geneToSubmit
- Current Background: Homo sapiens
- 1733 DAVID IDs
- Check Defaults
- Clear All

Below this, there's a list of selected categories with checkboxes:

- Disease (2 selected)
- Functional_Annotations (5 selected)
- Gene_Ontology (3 selected)
- General_Annotations (0 selected)
- Interactions (1 selected)
- Literature (0 selected)
- Pathways (3 selected)

Under 'Pathways', there's a table and a chart:

	5.1%	89	Chart
<input checked="" type="checkbox"/> BBID	5.1%	89	Chart
<input checked="" type="checkbox"/> BIOCARTA	15.8%	273	Chart
<input type="checkbox"/> EC_NUMBER	26.9%	467	Chart
<input checked="" type="checkbox"/> KEGG_PATHWAY	50.4%	874	Chart
<input type="checkbox"/> REACTOME_PATHWAY	64.3%	1115	Chart
<input type="checkbox"/> WIKIPATHWAYS	55.0%	953	Chart

At the bottom of the pathway list, there are two more categories:

- Protein_Domains (4 selected)
- Tissue_Expression (0 selected)

Dalla Home Page:

Il Database for Annotation, Visualization and Integrated Discovery (DAVID) fornisce una serie completa di **strumenti di annotazione funzionale** che consentono ai ricercatori di **comprendere il significato** biologico di **grandi liste di geni**. Questi strumenti sono alimentati dalla Knowledgebase DAVID, costruita sul concetto di gene DAVID, che riunisce diverse fonti di annotazioni funzionali.

Nello specifico, è stato da noi utilizzato per visualizzare i geni con le KEGG pathway maps.

Pathway Analysis

- La **pathway analysis** è un insieme di **strumenti** usato nell'ambito di ricerca delle scienze biologiche con il fine ultimo di **dare un significato** agli high-throughput biological data (HTBD)
- Lo scopo principale degli strumenti di PA è di **analizzare i dati** ottenuti dalle tecnologie che producono gli HTBD (quali RNA-microarrays, tandem mass spectrometry e RNA-sequencing) ed **individuare i gruppi relativi ai geni** legati alle alterazioni nei campioni di caso rispetto ad un campione di controllo.
- Ciò è stato ottenuto sulla base dell'**accoppiamento delle conoscenze** biologiche esistenti provenienti da banche dati con **test statistici, analisi matematiche e algoritmi computazionali**.
- I metodi di PA hanno aiutato i ricercatori nell'**identificazione dei ruoli biologici dei geni candidati**, selezionati per progettare nuove terapie per il cancro, aggirando i danni collaterali alle cellule sane.



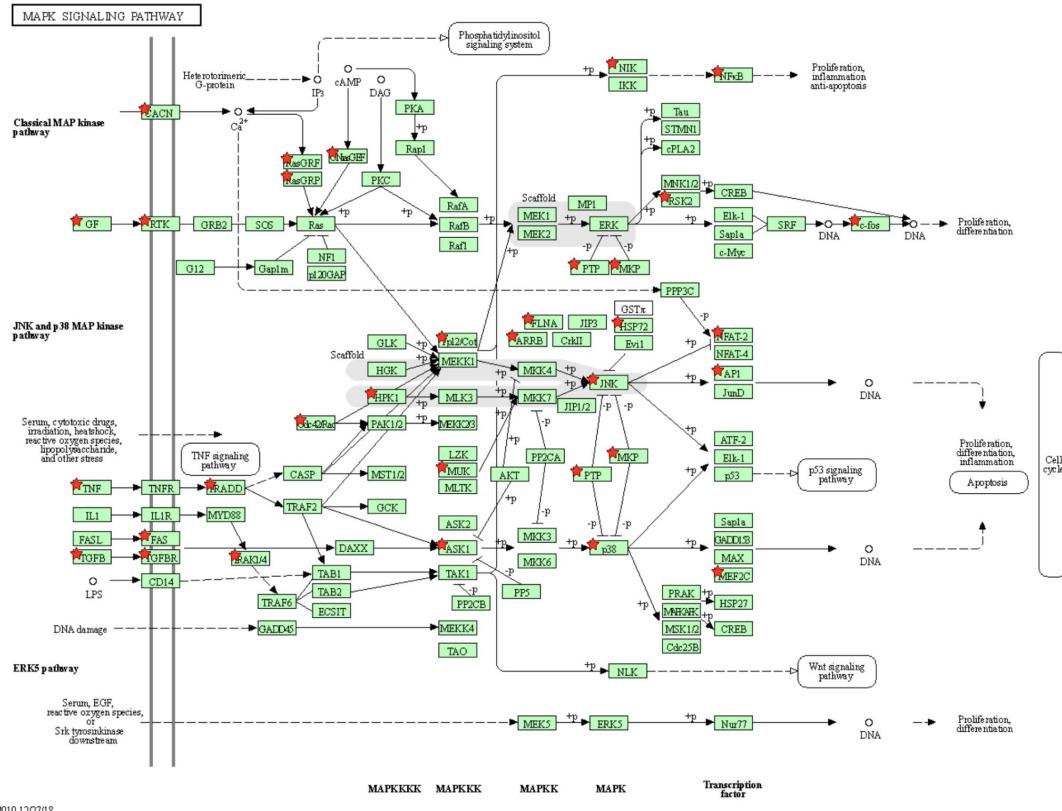
PDB: Kyoto Encyclopedia of Genes and Genomes

- Dalla home page:

“KEGG is a database resource for **understanding high-level functions** and utilities of the biological system, such as the cell, the organism and the ecosystem, from molecular-level information, especially large-scale molecular datasets generated by genome sequencing and other high-throughput experimental technologies.”

- Nello specifico, noi ci siamo avvalsi della **KEGG Pathway** che è una **collezione di pathway maps disegnate a mano** che rappresentano la **conoscenza allo stato dell'arte dell'interazione molecolare**, delle reazioni e dei relation networks per:
 - Metabolismo
 - Elaborazione delle informazioni genetiche
 - Processi cellulari
 - Sistemi organici
 - Malattie dell'uomo
 - Sviluppo di farmaci

Cancer biological Pathway (DAG)



Un esempio di pathway map del KEGG database.

In particolare, questa mappa rappresenta la **MAPK signaling pathway**, che è coinvolta nella regolazione dell'espressione genica e nello sviluppo e nella sopravvivenza delle cellule.

I geni contrassegnati dalle stelle rosse, sono i geni da noi rilevati in fase di validazione biologica.

Output Files (1/2)

Sublist	Category	Term	RT	Genes	Count	%	P-Value	Benjamini
	KEGG_PATHWAY	ECM-receptor interaction	RT	■	37	1,8	1,9E-13	6,1E-11
	KEGG_PATHWAY	Human papillomavirus infection	RT	■	79	3,9	7,9E-12	1,3E-9
	KEGG_PATHWAY	Focal adhesion	RT	■	56	2,8	2,7E-11	2,9E-9
	KEGG_PATHWAY	Pathways in cancer	RT	■	104	5,1	7,9E-10	6,5E-8
	KEGG_PATHWAY	Proteoglycans in cancer	RT	■	51	2,5	1,6E-8	9,9E-7
	KEGG_PATHWAY	Axon guidance	RT	■	47	2,3	1,8E-8	9,9E-7
	KEGG_PATHWAY	Regulation of actin cytoskeleton	RT	■	54	2,7	3,9E-8	1,8E-6
	KEGG_PATHWAY	Wnt signaling pathway	RT	■	43	2,1	1,5E-7	6,2E-6
	KEGG_PATHWAY	Protein digestion and absorption	RT	■	31	1,5	2,1E-7	6,9E-6
	KEGG_PATHWAY	Arrhythmogenic right ventricular cardiomyopathy	RT	■	26	1,3	2,2E-7	6,9E-6
	KEGG_PATHWAY	PI3K-Akt signaling pathway	RT	■	71	3,5	2,3E-7	6,9E-6
	KEGG_PATHWAY	Cell adhesion molecules	RT	■	37	1,8	7,5E-6	2,0E-4
	KEGG_PATHWAY	Adherens junction	RT	■	22	1,1	1,1E-5	2,7E-4
	KEGG_PATHWAY	Tight junction	RT	■	38	1,9	1,7E-5	4,0E-4
	KEGG_PATHWAY	Hippo signaling pathway	RT	■	36	1,8	1,9E-5	4,1E-4
	KEGG_PATHWAY	Leukocyte transendothelial migration	RT	■	29	1,4	2,0E-5	4,1E-4
	KEGG_PATHWAY	Breast cancer	RT	■	33	1,6	7,1E-5	1,4E-3
	KEGG_PATHWAY	Basal cell carcinoma	RT	■	19	0,9	7,5E-5	1,4E-3
	KEGG_PATHWAY	Small cell lung cancer	RT	■	24	1,2	8,1E-5	1,4E-3
	KEGG_PATHWAY	Rap1 signaling pathway	RT	■	42	2,1	1,0E-4	1,7E-3
	KEGG_PATHWAY	TGF-beta signaling pathway	RT	■	24	1,2	1,4E-4	2,1E-3
	KEGG_PATHWAY	Hypertrophic cardiomyopathy	RT	■	23	1,1	1,6E-4	2,4E-3
	KEGG_PATHWAY	AGE-RAGE signaling pathway in diabetic complications	RT	■	24	1,2	3,1E-4	4,4E-3
	KEGG_PATHWAY	Human T-cell leukemia virus 1 infection	RT	■	42	2,1	3,6E-4	4,9E-3
	KEGG_PATHWAY	Parathyroid hormone synthesis, secretion and action	RT	■	24	1,2	7,5E-4	9,8E-3
	KEGG_PATHWAY	Hepatocellular carcinoma	RT	■	33	1,6	9,0E-4	1,1E-2
	KEGG_PATHWAY	Vascular smooth muscle contraction	RT	■	28	1,4	9,2E-4	1,1E-2
	KEGG_PATHWAY	Cushing syndrome	RT	■	31	1,5	9,8E-4	1,1E-2



Output Files (2/2)



Term	Count	Genes
hsa00010:Glycolysis / Gluconeogenesis	14	ENO3, LDHB, ALDH3B2, ADH4, ALDH1B1, ALDH3B1, ALDOC, GALM, ACSS1, FBP1, PFKM, FBP2, PFKP, PC
hsa00040:Pentose and glucuronate interconversions	10	UGDH, UGT1A10, AKR1B10, UGT1A1, SORD, AKR1B1, UGT1A3, UGT1A9, UGT1A8, UGT1A6
hsa00051:Fructose and mannose metabolism	10	AKR1B10, PFKFB3, SORD, AKR1B1, ALDOC, FBP1, PFKM, FBP2, PFKP, KHK
hsa00052:Galactose metabolism	8	GALE, AKR1B10, B4GALT1, SI, AKR1B1, GALM, PFKM, PFKP
hsa00053:Ascorbate and aldarate metabolism	8	UGDH, UGT1A10, UGT1A1, ALDH1B1, UGT1A3, UGT1A9, UGT1A8, UGT1A6
hsa00140:Steroid hormone biosynthesis	17	UGT1A10, UGT1A1, AKR1C1, AKR1C3, AKR1C2, HSD17B6, HSD17B11, CYP19A1, HSD11B2, SULT1E1, H
hsa00260:Glycine, serine and threonine metabolism	10	GCSH, AOC3, GATM, BHMT, AOC2, DAO, PSAT1, SHMT1, AMT, PHGDH
hsa00590:Arachidonic acid metabolism	17	PLA2G2F, CBR1, PTGIS, PLA2G4D, HPGD, PLA2G4E, PLA2G2A, AKR1C3, PLA2G4A, PLA2G5, PTGR1, CYP2
hsa00601:Glycosphingolipid biosynthesis - lacto and neolactose	7	FUT5, B4GALT1, FUT9, ST3GAL4, A4GALT, FUT1, ABO
hsa00650:Butanoate metabolism	7	HMGCS1, ACSM3, OXCT1, OXCT2, HMGCS2, ACADS, HMGCLL
hsa00790:Folate biosynthesis	8	CBR1, AKR1B10, SPR, ALPP, PAH, AKR1C3, GGH, AKR1B1
hsa00830:Retinol metabolism	14	UGT1A10, UGT1A1, HSD17B6, ADH4, CYP2B6, RDH11, RDH10, CYP1A1, RDH5, UGT1A3, DHRS4L2, UGT
hsa00980:Metabolism of xenobiotics by cytochrome P450	15	CBR1, UGT1A10, UGT1A1, AKR1C1, MGST1, ALDH3B2, ADH4, CYP2B6, ALDH3B1, CYP1A1, UGT1A3, UG
hsa01522:Endocrine resistance	19	JAG2, CDKN1A, JUN, JAG1, CDKN2A, ADCY2, FOS, ABCB11, DLL1, ADCY7, MAPK13, ADC
hsa03320:PPAR signaling pathway	17	HMGCS1, ADIPOQ, ACSL5, APOC3, SORBS1, CYP8B1, CPT1B, FABP4, FABP5, FABP6, PLIN4, ME1, PLIN2
hsa04010:MAPK signaling pathway	46	CSF1R, TGFA, CACNA1H, RASGRP3, FGFR5, CACNG7, MECOM, ERBB2, STMN1, RAC2, FLNA, MAP3K8, FLN
hsa04015:Rap1 signaling pathway	35	CSF1R, RGS14, ADCY4, LPAR2, PIK3R3, LPAR3, CALML3, ADCY2, THBS1, PRKCZ, ADCY7, ADCY5, RASGRP
hsa04020:Calcium signaling pathway	41	CHRM2, MYLK2, CHRNA7, CAMK2A, ADCY4, ATP2A3, HTR2C, CXCR4, ITPR2, CALML3, ADCY2, ITPR3, M5
hsa04022:cGMP-PKG signaling pathway	27	MYLK2, NPYR1, IRS1, PDE3B, ADCY4, ATP2A3, ITPR2, CALML3, ADCY2, ITPR3, ADRB2, ADCY7, MYLK, ADC
hsa04024:cAMP signaling pathway	37	CHRM2, NPYR1, PDE3B, CAMK2A, ADCY4, ATP2A3, PIK3R3, CALML3, ADCY2, ADRB2, ADCY7, ADCY5, CRT
hsa04060:Cytokine-cytokine receptor interaction	49	ACVR1L, CSF3, TNFRSF6B, AMHR2, MPL, CXCL17, CX3CL1, CXCL16, AMH, CCR8, IL11RA, LIFR, TGFBR2, I
hsa04068:FoxO signaling pathway	22	CDKN1A, HOMER1, GADD45A, IRS1, PLK2, INSR, PIK3R3, SLC2A4, SOD2, GADD45G, MAPK13, TGFBR2,
hsa04080:Neuroactive ligand-receptor interaction	62	NPFFR2, CHRM2, UCN, THR8, HTRA, HTR2C, ADM, TMEM97, GRIK2, GRM3, GRM2, UCN2, CTSG, PRSS3
hsa04115:p53 signaling pathway	18	CDKN1A, RRM2, CD82, CDKN2A, GADD45A, IGFBP3, THBS1, GADD45G, BBC3, CCNB1, TP53I3, CCND2,
hsa04145:Phagosome	32	ITGB5, TUBA13, TFR, C1R, NCF4, THBS2, CORO1A, THBS1, FCAR, COMP, TUBB8, TUBB6, HLA-DMA, MR
hsa04151:PI3K-Akt signaling pathway	55	CHRM2, CDKN1A, CSF3, ITGB5, LAMC3, IRS1, ITGB4, BRCA1, COMP, FGFR5, GNGT1, CCND2, CREB3L1, TI
hsa04210:Apoptosis	23	JUN, TUBA13, BCL2A1, GADD45A, CTSG, DAB2IP, PIK3R3, ITPR2, CTSW, ITPR3, CSF2RB, FOS, GADD45G,
hsa04218:Cellular senescence	32	CDKN1A, PIK3R3, ITPR2, CALML3, ITPR3, FOXM1, CCNB1, CCND2, RASSF5, E2F1, MYBL2, MAP2K6, CD1
hsa04270:Vascular smooth muscle contraction	32	MYLK2, RAMP2, CALCA, RAMP3, CALCB, NPYR1, ADCY4, ITPR2, ADM, CALML3, ADCY2, ITPR3, PLA2G5, AI
hsa04310:Wnt signaling pathway	28	CAMK2A, PRICKLE4, ZNRF3, WNT11, CCND2, FRZB, RSPO2, RAC2, WNT2, FZD1, JUN, FZD3, TLE2, MMP
hsa04360:Axon guidance	31	SEMA5B, CAMK2A, SEMA3G, CXCR4, CXCR4, PIK3R3, SEMA3E, SEMA3F, PRKCZ, ROBO1, PARD6B, PLXN
hsa04390:Hippo signaling pathway	27	PRKCZ, BBC3, PARD6B, WNT11, CCND2, AMH, WNT2, TEAD3, FZD1, WWTR1, FZD3, WNT5B, FZD5, WN
hsa04510:Focal adhesion	46	MYLK2, LAMA2, ITGB5, LAMC3, ITGB4, LAMA4, PIK3R3, THBS2, THBS1, MYLK, COMP, VTN, CCND2, ERB
hsa04512:ECM-receptor interaction	30	LAMA2, ITGB5, SDC4, LAMC3, ITGB4, LAMA4, HMMR, THBS2, THBS1, COMP, VTN, SV2A, SPP1, TNR, ITG
hsa04514:Cell adhesion molecules	39	CD274, SELPLG, NLGN2, SDC4, NRXN1, CD80, SDC2, LRRC4, ICAM2, PTPRM, VTCN1, F11R, CLDN1, PTPF

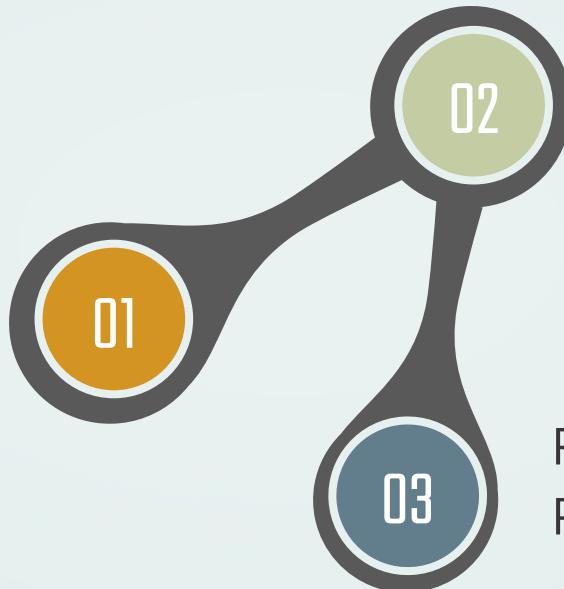


DISCUSSIONE dei RISULTATI

You could enter a subtitle here
if you need it

HIGHLIGHTS

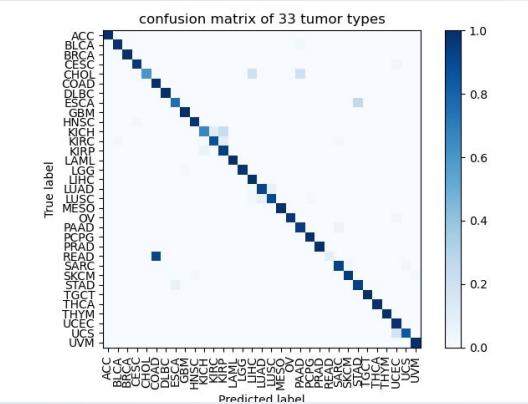
RISULTATI della
CLASSIFICAZIONE



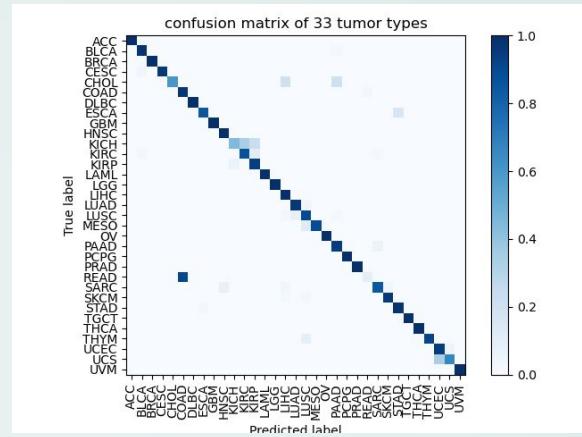
RISULTATI di GUIDED
GRAD-CAM

RISULTATI della
PATHWAY ANALYSIS

Risultati della Classificazione (GPU Net vs GPU VarNet)



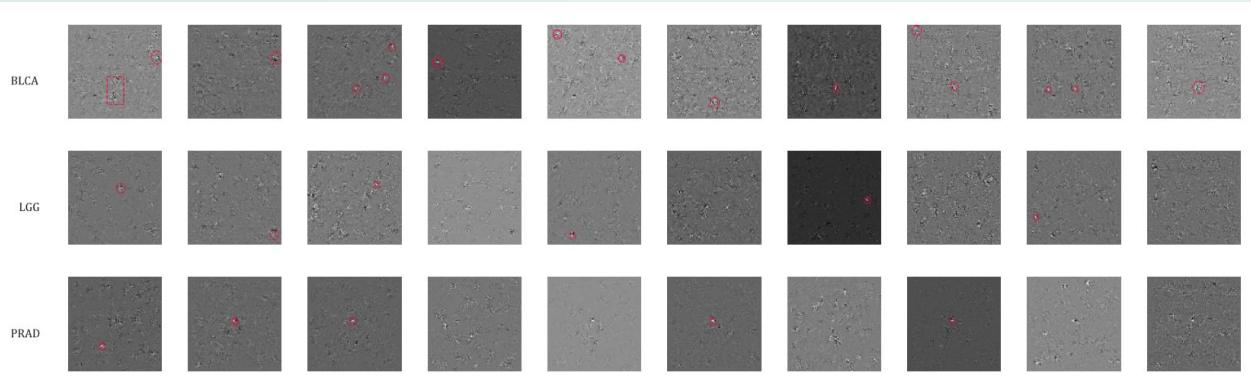
Net GPU



VarNet GPU

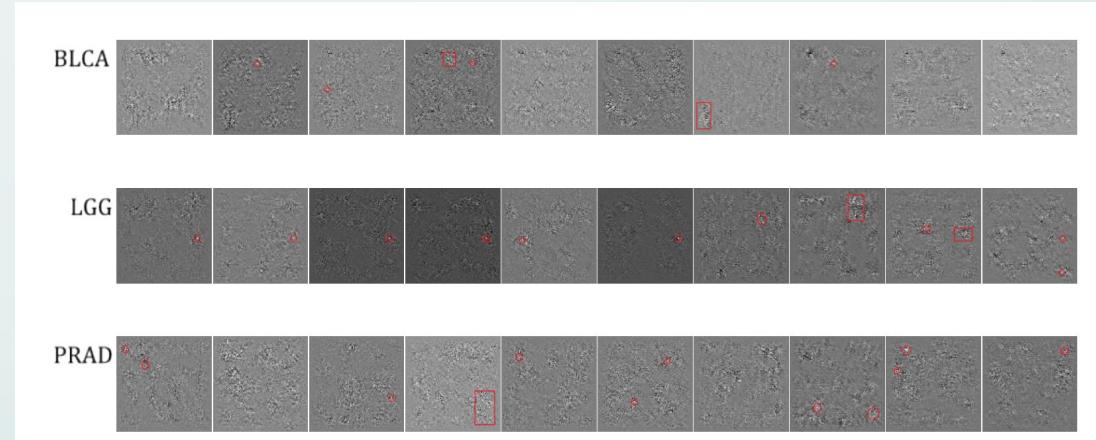
Modello	Accuracy	Precision	Recall	F1-score
Net	94.74%	95.10%	94.74%	94.37%
VarNet	94.83%	94.66%	94.83%	94.45%

Risultati Guided Grad-CAM



Net

VarNet



Biological Pathways Net vs VarNet

BRCA	hsa04915	Estrogen signaling pathway	2.38e-08
	hsa04512	ECM-receptor interaction	2.44e-08
	hsa04151	PI3K-Akt signaling pathway *	4.61e-06
	hsa04927	Cortisol synthesis and secretion	8.11e-06
	hsa04928	Parathyroid hormone synthesis, secretion and action	2.10e-05
	hsa04934	Cushing syndrome	3.31e-05
	hsa04510	Focal adhesion	4.96e-05
	hsa05205	Proteoglycans in cancer	8.05e-05
	hsa05165	Human papillomavirus infection	8.86e-05
	hsa03320	PPAR signaling pathway *	8.95e-05
	hsa04514	Cell adhesion molecules	1.03e-04
	hsa04145	Phagosome	1.20e-04
	hsa04933	AGE-RAGE signaling pathway in diabetic complications	1.51e-04
	hsa05200	Pathways in cancer	5.14e-04
	hsa01522	Endocrine resistance	7.22e-04
	hsa04926	Relaxin signaling pathway	7.76e-04
	hsa04540	Gap junction	9.62e-04

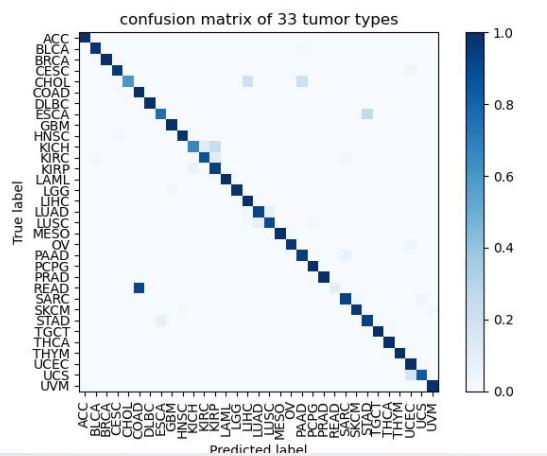
Le due reti hanno portato ad avere i medesimi risultati



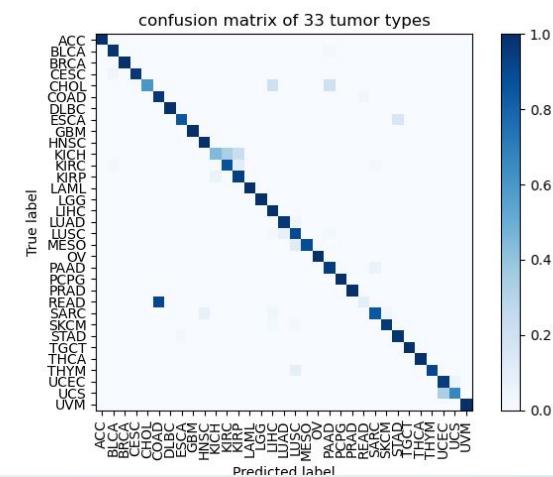
09

CONCLUSIONI e SVILUPPI FUTURI

Net vs VarNet



- Dalla **matrice di confusione** si può notare che la maggior parte delle classi sono classificate correttamente, tuttavia ci sono alcune classificazioni errate:
 - I campioni **READ** sono stati per lo più **erroneamente classificati** in **COAD** e ciò potrebbe essere dovuto ai pochi campioni di READ rispetto a quelli di COAD;



- Alcuni campioni di **ESCA** sono classificati erroneamente come **STAD**, questo potrebbe essere causato da alcuni campioni di ESCA i quali rientrano nella tipologia tumorale di carcinoma e che vengono visti dal modello come adenocarcinomi.
- Alcuni campioni di **UCS** sono classificati erroneamente come **UCEC** e ciò può essere causato dalla difficoltà del **modello a discriminare correttamente una particolare caratteristica** che emerge nei campioni UCS: essi sono legati ad un tipologia particolare di cancro cioè il carcinosarcoma, questo significa, in ambito biologico, che osservando i campioni di tale coorte tumorale al microscopio si vede che a livello di proprietà istologiche, essi mostrano caratteristiche sia del tumore carcinoma endometriale sia del sarcoma.

Conclusioni

- In questo progetto, il nostro obiettivo era quello di replicare il lavoro fatto B. Lyu e Haque nel loro paper **“Deep learning based tumor type classification using gene expression data”**.
(In Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics. 89–96).
- Impresa resa difficile dalla **scarsa documentazione del codice**. Di fatti, dapprima abbiamo scritto le parti di codice mancanti e poi ci siamo occupati dell'esecuzione dei test
- Successivamente, abbiamo provato a **migliorare i risultati** usando una **variante** alla rete convoluzionale da loro proposta
- Abbiamo ottenuto, in entrambi i casi, risultati migliori sia per le performance generali sia per le accuracy riferite a singole coorti **tumorali specifiche**
- Inoltre, in sede di validazione biologica, abbiamo ottenuto delle **nuove pathway** rispetto al paper di riferimento (dovuto **all'aggiornamento** dei **database** biologici negli ultimi 5 anni)

Sviluppi futuri

- Provare a migliorare ulteriormente le performance utilizzando un modello di deep learning diverso:
 - Ad esempio, reti CNN pre-trained quali VGG, AlexNet o Inception
 - Transformers
- Oppure cambiare paradigma di approccio utilizzando:
 - Reinforcement learning
 - Federated Learning
 - Combinare diversi paradigmi (ad es: Deep Learning + Reinforcement learning)

GRAZIE per l'ATTENZIONE

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), and infographics & images by [Freepik](#)

Please keep this slide for attribution

