# Leveraging gene expression and genomic varation for cancer prediction using one-shot learning

Alberto Montefusco, Alessandro Macaro
A.A. 2023/2024

**Abstract**

In questo lavoro si è inizialmente proceduto con una notevole espansione del dataset al fine di potenziare la diversità e la rappresentatività dei dati. Tale adeguamento ha consentito al modello di apprendere da una gamma più ampia di esempi, migliorando sensibilmente la sua capacità di generalizzazione. In aggiunta, sono state apportate modifiche a entrambi i modelli coinvolti nello studio, ossia il modello di classificazione e quello di tipo siamese.
Infine, si è introdotta una specifica riguardante le tipologie di mutazioni. Questa strategia ha contribuito non solo a raffinare la capacità del modello nel riconoscere i tipi di cancro, ma ha anche favorito una comprensione più approfondita dei dati genetici sottostanti. L'insieme di tali raffinamenti ha notevolmente potenziato l'efficacia e la solidità dei modelli, aprendo prospettive promettenti per la loro applicazione in ambiti diversificati all'interno del contesto dell'analisi genetica.

## 1 Introduzione

Sono trascorsi più di 20 anni dal completamento della bozza della sequenza del genoma umano. Questo traguardo ha portato alla generazione di una straordinaria quantità di dati genomici.
Il sequenziamento del DNA e altre tecniche biologiche continueranno ad aumentare il numero e la complessità di questi set di dati. A fronte dell'incremento della complessità dei dati genomici, sempre più ricercatori si affidano a strumenti computazionali basati su AI/ML in grado di gestire, estrarre e interpretare le preziose informazioni nascoste in questa grande mole di dati, con finalità sia di ricerca che di promozione della salute.

### 1.1 Stato dell'arte

The Cancer Genome Atlas (TCGA), un programma di riferimento per la genomica del cancro, ha caratterizzato molecolarmente oltre 20.000 campioni primari di cancro e campioni normali appaiati che coprono 33 tipi di cancro. Questo sforzo congiunto tra l'NCI e il National Human Genome Research Institute è iniziato nel 2006.
Nei dodici anni successivi, il TCGA ha generato oltre 2,5 petabyte di dati genomici, epigenomici, trascrittomici e proteomici. Questi dati hanno portato a miglioramenti nella capacità di diagnosticare, trattare e prevenire il cancro contribuendo a stabilire l'importanza della genomica del cancro.

### 1.2 Contributo di questo lavoro

Durante il processo sperimentale, è stata notevolmente aumentata la dimensione del dataset impiegato al fine di migliorare la diversità e la rappresentatività dei dati. Questo adeguamento ha consentito al modello di apprendere da una più ampia varietà di esempi, migliorando la sua generalizzazione.
Inoltre, sono state apportate modifiche a entrambi i modelli coinvolti in questo studio, sia il modello di classificazione che quello di tipo siamese. Un elemento chiave dell'ottimizzazione è stata l'implementazione dell'uso di pesi personalizzati. Questa strategia ha permesso di attribuire un peso differenziato alle diverse istanze del dataset in base alla quantità di sample presenti.
Infine, è stata introdotta una specifica riguardante le tipologie di mutazioni, ciò ha consentito una maggiore precisione nell'analisi delle informazioni genetiche.

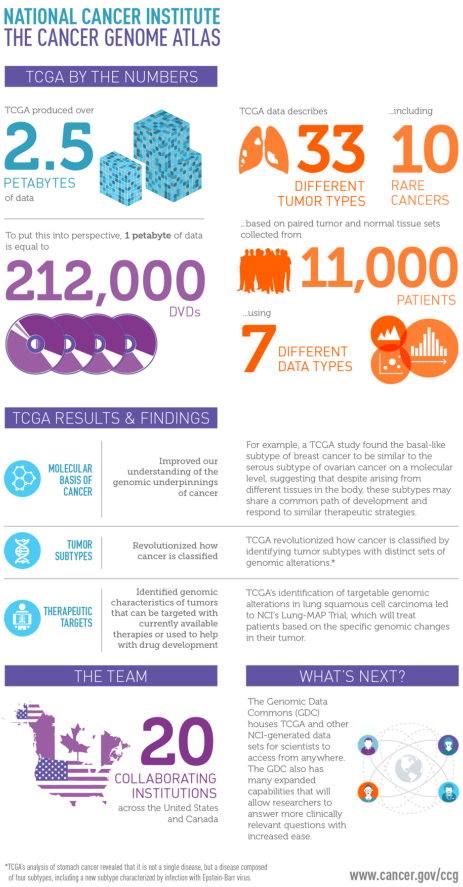


Figure 1: Infografica dei dati di TCGA

## 2 Lavori correlati

Nel panorama attuale della ricerca si stanno conducendo numerosi studi mirati all'individuazione di una firma genomica distintiva per diversi tipi di cancro.

In questa sezione verrano presentati papers che hanno delle similitudini con il nostro elaborato; sia per quanto riguarda l'obiettivo e sia per alcuni tool usati nel processo produttivo.

## 2.1 A deep learning system accurately classifies primary and metastatic cancers using passenger mutation patterns

Il presente paper analizza se sia possibile utilizzare tecniche di machine learning per determinare con precisione l'organo di origine e l'istologia del tumore utilizzando i modelli di mutazione somatica identificati dal sequenziamento del DNA dell'intero genoma.
Uno dei motivi principali di questo studio è stato quello di dimostrare la fattibilità di uno strumento diagnostico basato sul sequenziamento di nuova generazione (NGS) per l'identificazione del tipo di tumore.
Grazie alla sua stabilità, dal DNA è particolarmente facile recuperare campioni di tumore freschi e storici; inoltre, poiché le mutazioni si accumulano nel DNA, formano un registro storico dell'evoluzione del tumore non influenzato dall'ambiente locale e metastatico.
Vengono usate tecniche di deep-learning per capire se un semplice protocollo di sequenziamento e analisi del DNA per la determinazione del tipo di tumore possa essere un utile complemento alle tecniche istopatologiche esistenti.

### 2.1.1 Workflow

Utilizzando il dataset Pan-cancer Analysis of Whole Genomes (PCAWG), sono state condotte analisi sulle mutazioni genomiche in diversi tipi di tumori. Sono state identificate e selezionate specifiche feature mutazionali, che sono state utilizzate per il training di una serie di classificatori. Questi classificatori sono stati addestrati per predire il tipo di tumore a partire dal profilo mutazionale.
Per ogni categoria di feature, è stato implementato un classificatore di tipo Random Forest (RF classifier). L'input di questi classificatori è rappresentato dal profilo delle caratteristiche mutazionali, mentre l'output di tali classificatori è una stima della probabilità che un campione di tessuto analizzato appartenga a un determinato tipo di tumore.

### 2.1.2 Metodologia

Per ciascuno dei 24 tipi di cancro selezionati dal set di campioni PCAWG, è stato prima utilizzato un modello Random Forest per addestrare i classificatori per ogni tipo di cancro. Successivamente è stata utilizzata una feed-forward neural network per la classificazione dei 24 tipi di cancro, basata su SNV e sulla distribuzione delle mutazioni. La funzione softmax viene utilizzata come funzione di attivazione nello strato di uscita della rete neurale che può essere interpretata come una distribuzione di probabilità dei 24 tipi di cancro. Il tipo di tumore previsto è stato selezionato prendendo il tipo con la maggiore probabilità.

### 2.1.3 Risultati

Questo classificatore raggiunge un'accuratezza del 91% sui campioni di tumore resistenti dell'88% e 83% rispettivamente sui campioni primari e metastatici indipendenti. Sorprendentemente, l'aggiunta di informazioni sulle mutazioni ha ridotto l'accuratezza.

## 2.2 Predicting and characterizing a cancer dependency map of tumors with deep learning

Il seguente paper analizza quanto sia ancora difficile collegare le dipendenze tumorali alle composizioni molecolari delle cellule tumorali o alle linee cellulari non esaminate e ulteriormente ai tumori.
Viene presentato DeepDEP, un modello di deep learning che predice le dipendenze tumorali utilizzando profili genomici integrati. Utilizza un esclusivo unsupervised pretraining che cattura le rappresentazioni genomiche tumorali non etichettate per migliorare l'apprendimento delle dipendenze dal cancro.

### 2.2.1 Workflow

Sono stati utilizzati i dati dal dataset TCGA (The Cancer Genome Atlas), in particolar modo i file riguardanti mutazioni del DNA, espressione genica, metilazione del DNA e alterazione del numero di copie (CNA).
DeepDEP è progettato per prevedere il punteggio dell'effetto genico di una dipendenza di interesse (DepOI), per un campione di cancro (CCL o tumore). È composto da codificatori per ogni tipo di dati genomici di un campione e dall'impronta digitale di un DepOI. Il modello completo è stato addestrato e testato utilizzando i dati CCL di Broad DepMap.

### 2.2.2 Metodologia

DeepDEP incorpora un design di apprendimento per trasferimento con un unsupervised pretraining utilizzando i campioni tumorali non etichettati per apprendere le rappresentazioni dei dati, seguito da una messa a punto dei parametri su campioni CCL etichettati per catturare la relazione tra genomica e dipendenze geniche.
Il modello è composto da:

- Autoencoder, per ciascun dato molecolare (mutazioni del DNA, espressione genica, metilazione del DNA e alterazioni del numero di copie (CNA))
- Rete di codifica, per astrarre le impronte digitali funzionali di una dipendenza genica di interesse (DepOI)
- Rete di predizione, per convertire le caratteristiche apprese in un punteggio di dipendenza

L'autoencoder è un modello di riduzione delle dimensioni nell'ambito del deep learning, preallenato per ciascun tipo di dati genomico attraverso l'utilizzo di 8238 campioni del TCGA, con un processo di ottimizzazione degli iperparametri.

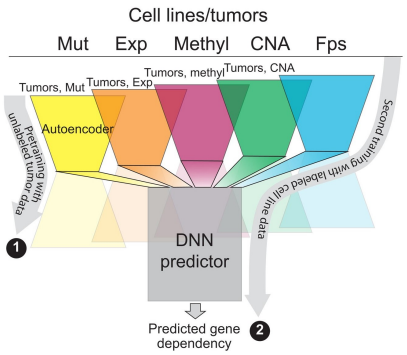


Figure 2: Workflow di DeepDEP

### 2.2.3 Risultati

Sono state confrontate le prestazioni del modello con variazioni di DeepDEP e ai metodi di machine learning convenzionali. In particolare, DeepDEP ha ottenuto un netto miglioramento delle prestazioni e della stabilità rispetto a un modello deep learning identicamente configurato senza unsupervised training.
I dati suggeriscono che le rappresentazioni non supervisionate di dati genomici apprese da grandi campioni di tumori sono vantaggiose per previsione di schermi di linee cellulari relativamente limitati.

## 2.3 CancerSiamese: one-shot learning for predicting primary and metastatic tumor types unseen during model training

Il seguente paper prende in considerazione la classificazione del cancro basata su dati di espressione genica utilizzando il Machine Learning. Grazie a iniziative come il TCGA, oggi la strategia prevalente è quella di addestrare un classificatore utilizzando campioni di tumore con tipi di cancro annotati. Gli sforzi precedenti si limitavano a classificare il cancro dai campioni normali utilizzando reti neurali. Sebbene abbiano riportato una buona accuratezza di classificazione, queste strategie hanno dei limiti che ne limitano l'adozione per la classificazione personalizzata dei tumori nell'era dell'oncologia di precisione:

- In primo luogo, l'elevata accuratezza della classificazione si basa sulla disponibilità di set di dati tumorali su larga scala ben annotati come il TCGA
- In secondo luogo, per tumori rari, non ci si potrebbe mai aspettare di raccogliere un numero sufficiente di campioni necessari per addestrare un modello con prestazioni soddisfacenti

Riconoscendo questi limiti della strategia attuale, viene proposta una nuova strategia di apprendimento one-shot, che richiede la raccolta di un solo campione "di supporto" per ogni nuovo tipo di cancro, un requisito drasticamente ridotto rispetto a quello tipico del TCGA con circa 500 campioni per tipo di cancro.

La classificazione viene effettuata confrontandolo con un insieme di campioni di supporto, uno per ogni tipo di cancro.
Viene proposto un modello SCNN, denominato CancerSiamese, che contiene due 1D-CNN identiche. Esse apprendono le rappresentazioni di tipo tumorale dei campioni di query e di supporto seguito da uno strato di apprendimento delle metriche per prevedere se le rappresentazioni del campione di riferimento e di quello di supporto sono simili o meno.

### 2.3.1 Workflow

Il dataset è composto da due set di dati:

- TCGA contiene 10.340 campioni provenienti da 33 diversi tipi di cancro primario
- MET500 comprende 765 campioni di 20 tipi di cancro di tumori metastatici

Come prima fase di pre-processing, tutti i tipi di tumore primario con un'origine simile sono stati raggruppati e poi rinominati con l'etichetta del tumore metastatico corrispondente in MET500. Dopo questo processo, il numero di tipi di cancro primario in TCGA è sceso da 33 a 29 (cioè quattro gruppi sono stati fusi). Dei 29 tipi, 9 sono unici per i tumori (primari) di TCGA, mentre 20 sono comuni tra TCGA e MET500 (primari e metastatici).
Dopodichè vengono filtrati i geni con media e deviazione standard inferiori a 0,8 in tutti i campioni, indipendentemente dal tipo di cancro, in TCGA e MET500.
Questa fase è stata concepita per ridurre l'effetto del rumore e dei geni che hanno uno scarso valore discriminatorio in questi due set di dati. Ciò ha ridotto il numero di geni in TCGA e MET500 a circa 6300 e 6200, rispettivamente, e sono stati determinati 4.858 geni comuni in entrambi i dataset.

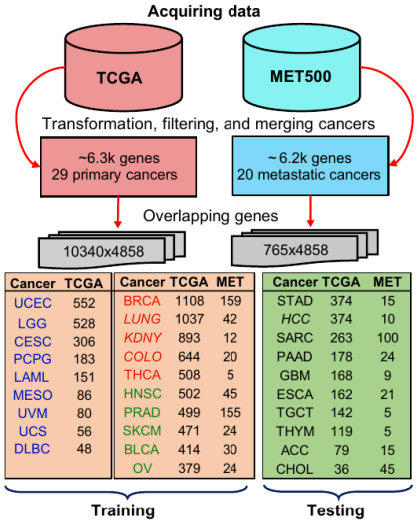


Figure 3: Workflow di pre-processing per l'estrazione dei campioni di tumori primari e metastatici per il training e il test

I dati di training comprendono campioni di 9 tipi di tumori primari unici e 10 tipi di tumori aggiuntivi con campioni primari e metastatici. I set di training sono stati selezionati in base al maggior numero di campioni primari o quando erano disponibili solo campioni TCGA. I dati di test includono i campioni primari e metastatici dei restanti 10 tipi di cancro. È importante notare che l'insieme di test non condivide alcun tipo di cancro con l'insieme di addestramento. Ciò è dovuto al fatto che l'obiettivo è quello di predire campioni di query da tipi di cancro non visti nell'insieme di addestramento.

### 2.3.2 Metodologia

CancerSiamese consiste in due CNN identiche applicate alla query e al campione di supporto, seguite da una metrica di somiglianza. Viene adottata la 1D-CNN grazie alla sua semplicità e all'elevata accuratezza nella predizione del tipo di cancro TCGA.
Dopo il primo strato di convoluzione 1D, vengono aggiunti due strati consecutivi di convoluzione 1D e maxpooling e infine uno strato di appiattimento. Analogamente viene scelta la funzione di attivazione relu per i primi due strati di convoluzione 1D e sigmoide per l'ultimo strato di convoluzione 1D. Per la metrica di similarità viene applicata una distanza L2 element-wise ai due vettori di feature generati dalle 1D-CNN e l'output è stato passato a due strati consecutivi completamente connessi seguiti da un nodo sigmoide per determinare se le coppie di input appartengono allo stesso tipo di cancro.

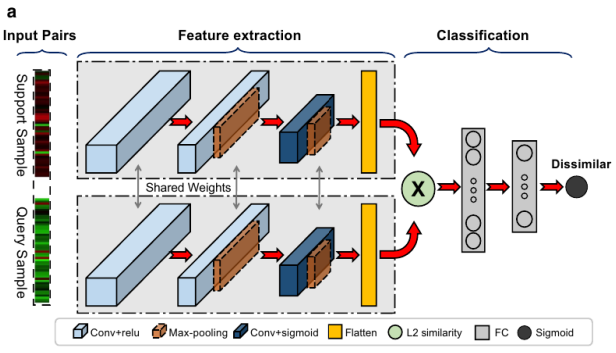


Figure 4: Architettura CancerSiamese

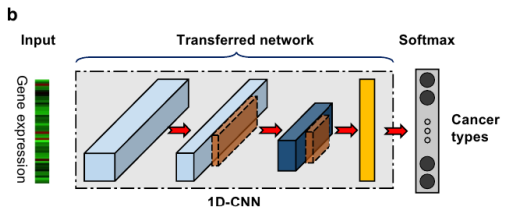


Figure 5: Architettura 1D-CNN, pre-addestrata per la classificazione dei tumori per inizializzare la parte di estrazione delle feature di CancerSiamese

### 2.3.3 Risultati

Le reti CancerSiamese per la predizione dei tumori primari sono state addestrate utilizzando tre diversi set di tipi di tumore primari, dove il numero totale di tipi di cancro era rispettivamente 9, 14 o 19.
Al contrario, è stata addestrata una sola rete CancerSiamese per la predizione dei tumori metastatici, utilizzando campioni di 10 tipi di tumore metastatico.

Viene utilizzato un classificatore 1-NN come modello di riferimento per confrontare le prestazioni dei modelli. Il classificatore 1-NN calcola la distanza euclidea tra l'espressione genica del campione di riferimento e quella del campione di supporto e seleziona l'etichetta con la distanza minima dal campione di riferimento come tipo predetto.
Le reti CancerSiamese addestrate sono state testate per diverse N-way Predictions (N = 6, 8 e 10). Per una N-way Predictions la rete CancerSiamese ha confrontato un campione di query con un set di supporto di N campioni, ciascuno di un diverso tipo di cancro. Il tipo del campione è stato predetto come il tipo del campione di supporto accoppiato se la coppia ha ricevuto la probabilità più alta da parte di CancerSiamese tra le N coppie.

Il modello addestrato con 19 tipi di cancro ha ottenuto le prestazioni più elevate con l'89,67%, l'87,32% e l'84,59% di accuratezza per le rispettive N-way Predictions.

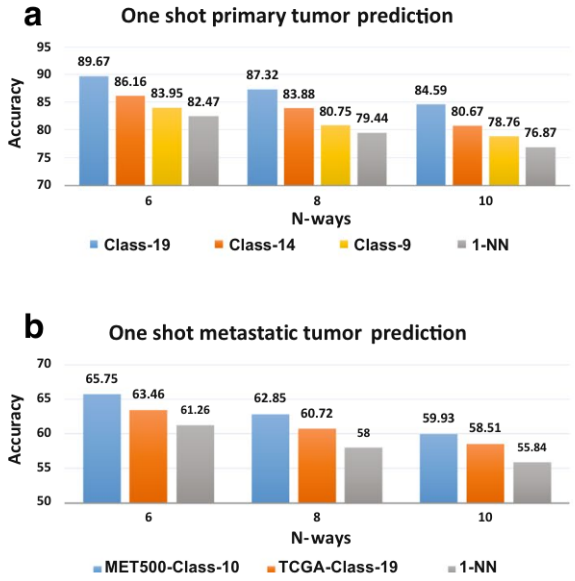


Figure 6: **a** Accuracy per i tipi di tumore primari; **b** Accuracy per i tipi di tumore metastatici

## 3 Background

In questa sezione verranno fornite alcune definizioni per capire meglio le tecniche e gli strumenti utilizzati che verranno citati nei capitoli successivi.

### 3.1 Siamese Neural Network

L'algoritmo per la rete neurale siamese è stato introdotto per la prima volta da Bromley e altri per rilevare firme false nel 1994. Prima di allora, Baldi e Chauvin avevano introdotto una rete neurale artificiale simile in grado di riconoscere le impronte digitali, anche se con un nome diverso.
Nello studio di Bromley, confrontando due firme, questa rete neurale siamese è stata in grado di stabilire se le due firme erano entrambe originali o se una era un falso.
Una rete neurale artificiale è un modello di apprendimento automatico composto da diversi strati di neuroni, che sono unità di elaborazione ispirate a neuroni biologici. Le tipiche reti neurali supervisionate sono chiamate *feed-forward* e si basano sul modello del percettrone. In una rete neurale feedforward, ogni neurone del primo strato legge un valore reale in ingresso, lo moltiplica per un peso e invia il risultato a tutti i neuroni dello strato successivo.
Nelle comuni rappresentazioni delle reti neurali, gli scienziati considerano lo strato più a sinistra come il primo strato e quello più a destra come lo strato di uscita. I neuroni di ogni strato successivo a quello di ingresso svolgono lo stesso lavoro e inviano i risultati ai neuroni dello strato successivo, fino allo strato finale (il processo va da sinistra a destra).
In una rete neurale supervisionata, lo strato finale invia i suoi risultati a uno strato di uscita a un neurone che produce il risultato della rete neurale. Durante il training, la rete neurale confronta i valori della rete neurale con il corrispondente valore e calcola l'errore statistico (di solito l'errore quadratico medio o l'errore di cross-entropia). In seguito, la rete neurale invia l'errore agli strati precedenti e aggiorna i pesi dei neuroni di conseguenza, attraverso la tecnica chiamata *error-back propagation*.

Il training si interrompe quando la rete neurale raggiunge il numero massimo di iterazioni impostato inizialmente. Una volta addestrato, il modello può essere applicato al test set: la rete neurale elaborerà ogni istanza dei dati di test (solo una volta, dal primo strato a sinistra all'ultimo strato a destra) e genererà un valore.

Le reti neurali feedforward con back-propagation dell'errore vengono utilizzate per le reti neurali siamesi. L'architettura della rete neurale siamese, infatti, contiene due reti neurali feedforward identiche unite all'uscita, che lavorano parallelamente. Durante il training, ogni rete neurale legge un profilo composto da valori reali e ne elabora i valori a ogni strato. La rete neurale attiva alcuni neuroni in base a questi valori, aggiorna i pesi attraverso la error-back propagation, e alla fine genera un profilo di uscita che viene confrontato con l'uscita dell'altra rete neurale. L'algoritmo confronta l'output della rete neurale superiore e quello della rete neurale inferiore attraverso una metrica di somiglianza: una distanza coseno nel modello originale.
Tramite questa misura di somiglianza, la rete neurale determina se i due profili sono diversi (valore di somiglianza del coseno nell'intervallo $[-1, 0]$) o simili (valore di somiglianza del coseno nell'intervallo $[0, +1]$). L'algoritmo etichetta quindi l'istanza di dati come *positiva* nel primo caso, o come *negativa* nel secondo caso.

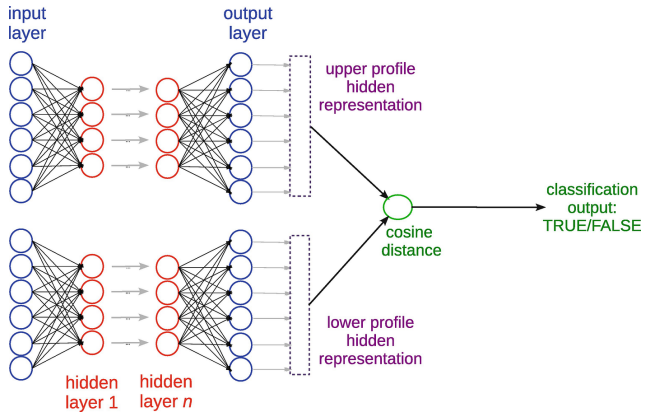


Figure 7: Siamese Neural Network

## 4 The Cancer Genome Atlas (TCGA)

The Cancer Genome Atlas (TCGA) ha raccolto molti tipi di dati per ciascuno degli oltre 20.000 campioni tumorali e normali. Ogni fase della pipeline di caratterizzazione del genoma ha generato numerosi dati, quali:

- informazioni cliniche, come lo stato di fumatore
- metadati relativi all'analisi molecolare, come il peso della porzione di campione
- dati di caratterizzazione molecolare, come i valori di espressione genica

I dati raccolti per un caso specifico nel TCGA possono differire in base alla qualità e alla quantità dei campioni, al tipo di tumore o alla tecnologia disponibile al momento dell'analisi.

### 4.1 TCGA Barcode

Il codice a barre TCGA è l'identificatore principale dei dati biospecifici nell'ambito del progetto TCGA. I codici a barre TCGA sono stati utilizzati per collegare tra loro i dati che attraversano la rete TCGA, poiché gli ID identificano in modo univoco una serie di risultati per un particolare campione prodotti da un particolare centro di generazione dei dati (ad esempio, GCC, GSC o GDAC).

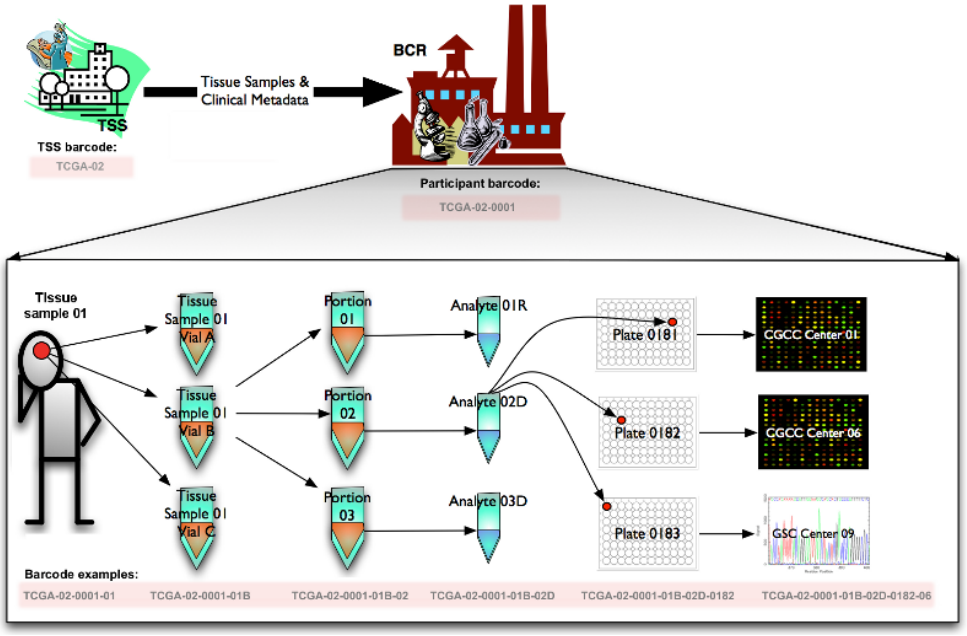


Figure 8: Creazione barcode TCGA

### 4.1.1 Creazione del TCGA Barcode

Tutti i codici a barre TCGA sono creati dal BCR.
La figura seguente illustra come un campione viene elaborato e gli viene assegnato un codice a barre TCGA in ogni fase. Partendo dal Tissue Source Site (TSS) e dal partecipante (che ha donato un campione di tessuto al TSS), vengono assegnati rispettivamente i codici a barre TCGA-02 e TCGA-02-0001.
Anche al campione stesso viene assegnato un codice a barre: TCGA-02-0001-01. Il campione viene suddiviso in fiale (ad es. TCGA-02-0001-01B) che vengono suddivise in porzioni (ad es. TCGA-02-0001-01B-02). Gli analiti (ad es. TCGA-02-0001-01B-02D) vengono estratti da ciascuna porzione e distribuiti in una o più piastre (ad es. TCGA-02-0001-01B-02D-0182), dove ogni pozzetto è identificato come un'aliquota (ad es. TCGA-02-0001-01B-02D-0182-06). Queste piastre vengono inviate a GCC o GSC per la caratterizzazione e il sequenziamento.

### 4.1.2 Leggere il TCGA Barcode

Un Barcode TCGA è composto da un insieme di identificatori. Ciascuno di essi identifica specificamente un dato per TCGA. La figura seguente illustra il modo in cui gli identificatori dei metadati compongono un codice a barre.

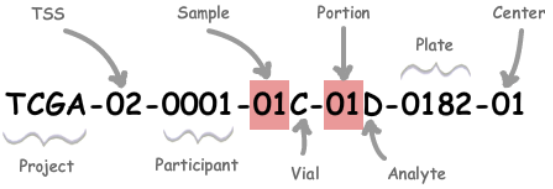


Figure 10: Esempio Barcode TCGA

| Label | Identificativo | Valore | Descrizione del valore | Possibili valori |
|---|---|---|---|---|
| **Analyte** | Tipo molecolare dell'analita da analizzare | D | L'analita è un campione di DNA | Vedere tabelle dei codici |
| **Plate** | Ordine della piastra in una sequenza di piastre da 96 pozzetti | 182 | La 182° piastra | Valore alfanumerico di 4 cifre |
| **Portion** | Ordine di porzione in una sequenza di porzioni di campione da 100 - 120 mg | 1 | La prima parte del campione | gen-99 |
| **Vial** | Ordine del campione in una sequenza di campioni | C | La terza fiala | A-Z |
| **Project** | Nome del progetto | TCGA | Progetto TCGA | TCGA |
| **Sample** | Tipo di campione | 1 | Un tumore solido | I tipi di tumore variano da 01 a 09, i tipi normali da 10 a 19 e i campioni di controllo da 20 a 29. Per un elenco completo dei codici dei campioni, consultare il rapporto sulle tabelle dei codici. |
| **Center** | Centro di sequenziamento o di caratterizzazione che riceverà l'aliquota per l'analisi | 1 | Il Broad Institute GCC | Vedere tabelle dei codici |
| **Participant** | Partecipante dello studio | 1 | Il primo partecipante del MD Anderson per lo studio sul GBM | Qualsiasi valore alfanumerico |

Figure 9: Tabella esplicativa barcode TCGA

## 5 Preprocessing dei dati

In questa sezione verrà esaminato in dettaglio l'intero processo che riguarda la selezione dei dati, la scelta delle feature e il preprocessing.

### 5.1 Data Collection

The Cancer Genome Atlas (TCGA) rappresenta il dataset di maggiore rilevanza in questo ambito, ma l'accesso ai dati in suo possesso è particolarmente ostico.
Per colmare questa difficoltà, è disponibile cBioPortal, una risorsa open-access e open-source che facilita l'esplorazione interattiva di dati genomici multidimensionali sul cancro. L'obiettivo di cBioPortal è abbattere significativamente le barriere tra dati genomici e i ricercatori nel campo del cancro, fornendo un accesso rapido, intuitivo e di alta qualità a profili molecolari e attributi clinici provenienti da progetti di genomica sul cancro su vasta scala. In particolare, cBioPortal mette a disposizione un vasto assortimento di progetti che incorporano frammenti del dataset TCGA.
Sono stati inclusi tutti i progetti che contengono sezioni di questo dataset, riguardanti una varietà di tipi di cancro, per un totale di 92 progetti.

### 5.2 Data Selection

Ciascun progetto comprende una varietà di file, suddivisi nelle seguenti categorie:

- **Clinici**, che comprendono
  - dati clinici, come informazioni demografiche, informazioni sul trattamento, dati sulla sopravvivenza
  - dati biospecifici, come informazioni sulle modalità di trattamento dei campioni
  - report patologici, solo per alcuni casi selezionati
- **Copy Number**, il cui file principale tratta del Low-Pass DNA Sequencing ovvero il sequenziamento dell'intero genoma di campioni tumorali e normali
- **DNA**, che include il sequenziamento dell'intero esoma di campioni tumorali e normali, nonchè il sequenziamento dell'intero genoma di campioni tumorali e normali (solo per alcuni casi selezionati)
- **Metilazione**, che include il sequenziamento dell'intero genoma eseguito dopo il trattamento con bisolfito dei campioni tumorali
- **mRNA Expression**, che include il file mRNA sequencing ovvero il sequenziamento di mRNA di campioni tumorali utilizzando una preparazione di RNA ad arricchimento di poli(A)
- **Protein Expression**, che include il file Reverse-PhaseArray Protein, il quale permette di studiare l'espressione delle proteine coinvolte nei processi cellulari coinvolti nella formazione e progressione dei tumori

Nel contesto di questo studio sono stati selezionati soltanto alcuni file tra tutti quelli disponibili, ovvero:

- *data_clinical_patient*, che contiene i dati clinici sul paziente (come Patient ID, sesso e stato del tumore)
- *data_clinical_sample*, che contiene i dati riguardanti i campioni tumorali
- *data_cna*, che contiene informazioni sulle variazioni nel numero di copie di segmenti specifici del DNA
- *data_methylation*, che contiene informazioni sulla metilazione del DNA
- *data_mrna_seq_v2_rsem*, che contiene il sequenziamento di mRNA di campioni tumorali
- *data_mutations*, che contiene i dati sulle mutazioni ottenuti tramite sequenziamento dell'intero esoma
- *data_rppa*, che contiene i dati sull'espressione delle proteine

Dei 92 progetti precedentemente inclusi, alcuni sono stati esclusi a causa dell'assenza di numerosi file essenziali per l'analisi condotta, mentre altri sono stati inclusi, vista la mancanza di un numero limitato di file. Questa prima fase ha ridotto il numero totale di progetti a 75.
La distribuzione dei sample relativi ai diversi tipi di cancro è la seguente:

| Tipo di Cancro | Organo di riferimento | Pazienti |
|---|---|---|
| Adrenocortical Carcinoma | Ghiandola surrenale | 79 |
| Breast Cancer | Seno | 1103 |
| Cervical Cancer | Cervice uterina | 306 |
| Hepatobiliary Cancer | Fegato e vie biliari | 409 |
| Colorectal Cancer | Colon e retto | 609 |
| Bladder Cancer | Vescica urinaria | 408 |
| Esophagogastric Cancer | Esofago e stomaco | 600 |
| Prostate Cancer | Prostata | 498 |
| Head and Neck Cancer | Testa e collo | 522 |
| Renal Cell Carcinoma | Rene | 891 |
| Leukemia | Midollo osseo e sangue | 173 |
| Non-Hodgkin Lymphoma | Tessuto linfatico | 48 |
| Non-Small Cell Lung Cancer | Polmone | 1014 |
| Mesothelioma | Membrana sierosa | 87 |
| Ovarian Epithelial Tumor | Ovaie | 300 |
| Pancreatic Cancer | Pancreas | 179 |
| Pheochromocytoma | Midollare del surrene | 147 |
| Melanoma | Pelle e occhi | 103 |
| Thyroid Cancer | Tiroide | 502 |
| Thymic Tumor | Timo | 120 |
| Ocular Melanoma | Occhi | 80 |
| Glioma | Sistema nervoso centrale (Cervello e midollo spinale) | 697 |
| Miscellaneous Neuroepithelial Tumor | Varie localizzazioni nel sistema nervoso centrale | 31 |
| Soft Tissue Sarcoma | Tessuti molli (muscoli, tessuto connettivo, ecc.) | 254 |
| Endometrial Cancer | Rivestimento dell'utero (endometrio) | 588 |
| Nerve Sheath Tumor | Sistema nervoso periferico | 9 |
| | | **9757** |

Figure 11: Distribuzione dei sample

### 5.3 Data Cleaning

Per ciascun progetto, sono state selezionate le feature più rilevanti dai file precedentemente selezionati. In questa sezione verrà fornito un elenco per ciascun file, accompagnato dalla relativa motivazione.

#### 5.3.1 Data_clinical_patient

All'interno del file *data_clinical_patient* sono presenti numerose informazioni riguardanti il paziente. Nel contesto di questo studio sono state selezionate solo le seguenti:

- *Age*, ovvero l'età alla quale è stato diagnosticato per la prima volta. La frequenza di molti tipi di cancro aumenta con l'età e alcuni tipi sono più comuni in determinate fasce d'età
- *Sex*, ovvero il sesso del paziente. Alcuni tipi di cancro sono più comuni in un sesso rispetto all'altro. Ad esempio, il cancro al seno è molto più comune nelle donne, mentre il cancro alla prostata è più comune negli uomini
- *OS status*, ovvero lo stato complessivo di sopravvivenza del paziente ed è un indicatore cruciale che fornisce informazioni chiave sulla progressione e l'andamento della malattia nel tempo
- *AJCC pathologic tumor stage*, ovvero l'estensione di un cancro, in particolare se la malattia si è diffusa dal sito originale ad altre parti del corpo in base ai criteri di stadiazione AJCC

#### 5.3.2 Data_clinical_sample

All'interno del file *data_clinical_sample* sono presenti numerose informazioni riguardanti il sample estratto dal paziente. Nel contesto di questo studio sono state selezionate solo le seguenti:

- *Sample ID*, ovvero l'identificatore del sample con riferimento al paziente. È importante perché un singolo paziente potrebbe aver fornito più campioni nel corso dello studio. Questi campioni possono provenire da diverse parti del corpo o da diversi momenti nel tempo
- *Cancer type*, ovvero il tipo di cancro. Ogni tipo di cancro ha caratteristiche biologiche uniche, che influenzano la sua crescita e diffusione
- *Sample type*, ovvero il tipo di campione (normale, primario, metastatico, recidiva). Identificare se il campione proviene da un tumore primario, da una metastasi o da una recidiva aiuta a capire come la malattia si è evoluta nel tempo e come è progredita attraverso il corpo
- *Somatic status*, viene utilizzato per descrivere le mutazioni genetiche o le alterazioni molecolari presenti nelle cellule tumorali di un individuo
- *Oncotree code*, ovvero un sistema di classificazione gerarchico utilizzato nel campo dell'oncologia per identificare e categorizzare i diversi tipi di tumori. Semplifica la ricerca e l'analisi dei dati oncologici

#### 5.3.3 Data_methylation

All'interno del file *data_methylation* sono contenute informazioni sull'espressione genica di diversi geni, misurati attraverso un metodo di analisi chiamato HM450, utilizzato per studiare i pattern di metilazione del DNA.
Le colonne del file rappresentano diverse categorie di dati:

- *Hugo Symbol*, ovvero un acronimo utilizzato per rappresentare un gene in modo univoco. Ogni gene ha uno Hugo Symbol unico
- le altre colonne rappresentano campioni diversi. Ogni colonna contiene valori che indicano l'espressione o i livelli di metilazione del gene corrispondente in ciascun campione

Questo file fornisce informazioni sulle modifiche al DNA (metilazioni), ovvero fornisce valori che indicano il grado di metilazione in specifiche regioni del DNA per ciascun gene. Ciò permette di analizzare le differenze nella metilazione del DNA tra campioni e identificare marcatori specifici di metilazione associati a determinati tipi di cancro.

#### 5.3.4 Data_mrna

Questo file ha una formattazione uguale al file precedentemente menzionato, ma c'è una differenza sul tipo di dati rappresentato.
Il file mRNA-seq fornisce informazioni sull'espressione genica, ossia quanto RNA messaggero (mRNA) è stato trascritto dai geni in un campione biologico.

#### 5.3.5 Data_rppa

Anche questo file ha una formattazione uguale al file precedentemente menzionato, ma ha come identificatore univoco *Composite.Element.REF*, utilizzato per le proteine o gli elementi compositi considerati nell'analisi proteomica.
Le misurazioni presenti nel file si riferiscono all'espressione delle proteine e sono state ottenute utilizzando un metodo chiamato "reverse-phase protein array" (RPPA). Il reverse-phase protein array è una tecnica analitica che consente di misurare l'espressione di specifiche proteine all'interno di campioni biologici.

### 5.3.6 Data_mutations

All'interno del file *data_mutation* sono contenuti dati sulle mutazioni ottenuti tramite sequenziamento dell'esoma completo.
Le colonne del file rappresentano diverse categorie di dati:

- *Hugo Symbol*, ovvero l'acronimo utilizzato per rappresentare un gene in modo univoco
- *Chromosome*, specifica il cromosoma sul quale è stata rilevata la mutazione
- *Start Position*, rappresenta la posizione di inizio della mutazione all'interno del cromosoma
- *End Position*, rappresenta la posizione di fine della mutazione all'interno del cromosoma
- *Strand*, indica il filamento del DNA coinvolto nella mutazione (+ per il filamento senso, - per il filamento antisenso)
- *Consequence*, descrive l'effetto o la conseguenza della mutazione (ad esempio, missense_variant indica una mutazione che cambia un singolo nucleotide)
- *Variant Classification*, classifica il tipo di variante (ad esempio, Missense_Mutation indica una mutazione che causa un cambiamento nell'aminoacido codificato)
- *Variant Type*, specifica il tipo di variante (ad esempio, SNP indica una variante a singolo nucleotide)
- *Tumor Sample Barcode*, identifica i campioni di tessuto tumorale prelevati dai pazienti
- *Matched Norm Sample Barcode*, identifica i campioni di tessuto normale (non tumorale) che sono stati prelevati dallo stesso paziente il cui tessuto tumorale è stato analizzato. Questi campioni di tessuto normale vengono utilizzati come controllo per confrontare le differenze genomiche e molecolari riscontrate nel tessuto tumorale

## 5.4 Data Transformation

Sono stati implementati script Python dedicati per automatizzare le operazioni di elaborazione dei file selezionati. Questo approccio è stato adottato per garantire una gestione ottimale del vasto insieme di dati e precisione nell'esecuzione delle operazioni.

### 5.4.1 Conversione da TXT a CSV

Ogni file di ciascun progetto è originariamente in formato txt. Tuttavia, dato che numerosi framework e librerie di Machine Learning, quali TensorFlow, scikit-learn e pandas, offrono strumenti per importare dati da file CSV, e considerando che questi dati possono essere elaborati con facilità utilizzando librerie come pandas, è stato deciso di convertire i file in formato CSV.

Per questo motivo, è stato sviluppato uno script per automatizzare il processo di conversione. La funzione Python chiamata *converti_in_csv* prende come argomento una directory e converte i file di testo con estensione txt in file CSV all'interno della directory specificata. Il ciclo for serve a iterare attraverso tutti i file presenti nella directory specificata.

```
def converti_in_csv(directory):
    for root, dirs, files in os.walk(directory):
        for filename in files:
```

Dopodichè si procede a controllare se l'estensione del file è txt tramite l'if.

```
if filename.endswith('.txt'):
    input_file_path = os.path.join(root, filename)
    output_csv_path = os.path.join(root,
    ↪  f'{os.path.splitext(filename[0]}.csv')
```

Se l'estensione del file è txt, il ciclo continua l'esecuzione e converte il file in un file CSV, separando i campi utilizzando il punto e virgola come delimitatore.

```
with open(input_file_path, 'r') as input_file,
open(output_csv_path, 'w', newline='') as
↪  output_csv:
    lines = input_file.readlines()
    csv_writer = csv.writer(output_csv,
    ↪  delimiter=';')
    for line in lines:
      csv_writer.writerow(line.strip().split('\t'))
```

### 5.4.2 Cambio delimitatore CSV

Dato che alcuni campi contengono virgole come parte del testo (ad esempio, nei file riguardanti i geni), utilizzare la virgola come delimitatore può creare problemi di interpretazione. Lo script Python *change_csv_delimiter* si occupa di risolvere questo problema.

```
input_file = '/blca_tcga/data_clinical_patient.csv'
output_file = '/blca_tcga/data_clinical_patient.csv'

with open(input_file, 'r', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter='\t')
    data = list(reader)

with open(output_file, 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=';')
    writer.writerows(data)
```

### 5.4.3 Data Cleaning Patient

Nel paragrafo precedente sono state menzionate le feature selezionate dai file *data_clinical_patient* e *data_clinical_sample*.
Lo script Python *data_cleaning_patient* si occupa di automatizzare la selezione di queste feature per tutti i file all'interno delle sottocartelle dei singoli progetti e di unirli in un unico file. Lo script è composto da due funzioni principali: *cerca_label_e_copia* e *elabora_sottocartelle*.

La prima funzione analizzata è *cerca_label_e_copia*. Per prima cosa, si imposta l'indice delle label da cercare a 4, dato che nei file menzionati le prime quattro righe contengono descrizioni delle label associate a ciascuna colonna. Successivamente, viene creata una lista che raccoglie i nomi delle colonne da individuare e copiare dal file CSV.

```
def cerca_label_e_copia(input_csv):
    indice_label = 4

    labels_da_cercare = ["SAMPLE_ID", "CANCER_TYPE",
    "SEX", "SAMPLE_TYPE", "SOMATIC_STATUS",
    "ONCOTREE_CODE", "OS_STATUS",
    "AJCC_PATHOLOGIC_TUMOR_STAGE", "AGE"]
```

Viene utilizzata la libreria pandas per leggere il file CSV specificato in input_csv e viene inizializzato un dizionario che verrà utilizzato per memorizzare le colonne corrispondenti alle label. Il ciclo for itera attraverso ogni label nella lista e viene controllato se la label è presente come nome di colonna nel DataFrame. Se la label è presente come colonna, la copia all'interno del dizionario.

```
df = pd.read_csv(input_csv, header=indice_label,
     delimiter=';')

colonne_da_copiare = {}

for label in labels_da_cercare:
    if label in df.columns:
        colonne_da_copiare[label] = df[label]
```

Viene creato un nuovo DataFrame utilizzando il dizionario, che conterrà solo le colonne corrispondenti alle label cercate, e restituisce il Dataframe risultante.

```
df_risultante = pd.DataFrame(colonne_da_copiare)

return df_risultante
```

La seconda funzione analizzata è *elabora_sottocartelle*. Il ciclo for serve a iterare attraverso tutti i file presenti nella directory specificata e verifica se i file data_clinical_patient e data_clinical_sample sono presenti nella lista dei file all'interno della directory corrente. Se i file sono presenti, vengono costruiti i path completi per entrambi i file.

```
def elabora_sottocartelle(directory_principale):
for root, dirs, files in
↪  os.walk(directory_principale):
    if 'data_clinical_patient.csv' in files and
    ↪  'data_clinical_sample.csv' in files:
        input_csv_patient = os.path.join(root,
        ↪  'data_clinical_patient.csv')

        input_csv_sample = os.path.join(root,
        ↪  'data_clinical_sample.csv')
```

Viene chiamata la funzione analizzata in precedenza per entrambi i file e memorizza i risultati nei DataFrame corrispondenti. Dopodichè i due DataFrame vengono concatenati lungo l'asse delle colonne.

```
df_patient = cerca_label_e_copia(input_csv_patient)
df_sample = cerca_label_e_copia(input_csv_sample)

df_concatenato = pd.concat([df_patient, df_sample],
↪  axis=1)
```

Viene creata una lista di label nell'ordine specificato e una lista che contiene solo le label che sono presenti come colonne nel DataFrame concatenato. Viene controllato se alcune delle label desiderate sono mancanti nel DataFrame concatenato. Se ci sono label mancanti, vengono identificate e memorizzate.

```
labels_ordine = ["SAMPLE_ID", "CANCER_TYPE", "SEX",
                "SAMPLE_TYPE", "SOMATIC_STATUS",
                "ONCOTREE_CODE", "OS_STATUS",
                "AJCC_PATHOLOGIC_TUMOR_STAGE",
                "AGE"]

labels_presenti = [label for label in labels_ordine
↪  if label in df_concatenato.columns]

if len(labels_presenti) < len(labels_ordine):
    label_mancanti = set(labels_ordine) -
    ↪  set(labels_presenti)
    print(f"Avviso: Alcune label non trovate nel
    ↪  DataFrame concatenato: {label_mancanti}")
```

Viene creato un nuovo DataFrame che contiene solo le colonne corrispondenti alle label presenti e il path completo per il file di output, all'interno della stessa directory della sottocartella corrente. Infine viene salvato il Dataframe risultante come un file CSV utilizzando il punto e virgola come delimitatore di campo.

```
df_risultante = df_concatenato[labels_presenti]

output_path = os.path.join(root, 'data.csv')
df_risultante.to_csv(output_path, sep=';',
↪  index=False)
print(f"Operazione completata per {root}. Output
↪  salvato in {output_path}.\n\n")
```

### 5.4.4 Data Cleaning Mutation

Anche in questo caso lo script Python *data_cleaning_mutation* si occupa di automatizzare la selezione delle feature scelte per il file data_mutation per tutti i file all'interno delle sottocartelle dei singoli progetti. Lo script è composto da due funzioni principali: *copia_colonne* e *modifica_files*.

La prima funzione analizzata è *copia_colonne*.
Viene creata una lista che contiene i nomi delle colonne che devono essere presenti nel file CSV di input. Si tenta di leggere il file CSV di input, specificando il delimitatore come punto e virgola (;), e imponendo che i dati vengano trattati come stringhe. Se il file non viene trovato, viene stampato un messaggio di errore.
Viene verificato se tutte le label necessarie sono presenti nel DataFrame caricato e crea una lista che contiene le label effettivamente presenti.

```
def copia_colonne(input_csv, output_csv):
    label_da_cercare = ["Hugo_Symbol",
    "Tumor_Sample_Barcode", "Chromosome",
    "Start_Position", "End_Position", "Strand",
    "Consequence", "Variant_Classification",
    "Variant_Type"]

    try:
        df = pd.read_csv(input_csv, delimiter=';',
        ↪  dtype='str', low_memory=False)
    except FileNotFoundError:
        print(f"Errore: Il file {input_csv} non è
        ↪  stato trovato.")
        return

    labels_presenti = [label for label in
    ↪  label_da_cercare if label in df.columns]
```

Se ci sono label mancanti vengono stampate a schermo e la funzione termina senza creare il nuovo file CSV.

```
labels_mancanti = set(label_da_cercare) -
↪  set(labels_presenti)
for label in labels_mancanti:
    print(f"Avviso: Label '{label}' non trovata nel
    ↪  CSV.")

if labels_mancanti:
    return
```

Se tutte le label sono presenti, vengono selezionate solo le colonne con le label desiderate e viene creato un nuovo DataFrame. Il DataFrame risultante veine scritto nel file CSV di output utilizzando il punto e virgola come delimitatore.

```
df_selezionato = df[labels_presenti]

df_selezionato.to_csv(output_csv, index=False,
↪  sep=';')
print(f"Operazione completata per {input_csv}.
↪  Output salvato in {output_csv}.")
```

La seconda funzione analizzata è *modifica_files*.
Questa itera i file e le directory presenti nella directory specificata e per ogni file trovato verifica se il nome del file corrisponde a "data_mutations.csv". In caso affermativo, compone i path di input e di output e chiama la funzione precedentemente analizzata passandoglieli.

```
def modifica_files(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file == 'data_mutations.csv':
                input_path = os.path.join(root,
                ↪  file)
                output_path = os.path.join(root,
                ↪  'data_mutations.csv')
                copia_colonne(input_path,
                ↪  output_path)
```

### 5.4.5 Normalizzazione

La normalizzazione è fondamentale nel pre-processing dei dati perchè serve a garantire che le variabili in un dataset abbiano una scala comune, in modo che il modello possa apprendere in modo efficace e convergere più rapidamente durante il training. Lo script Python *normalize* si occupa di automatizzare la normalizzazione nei file in cui occorre.
Esso è composto da due funzioni principali: *normalizza_e_concatena* e *elabora_sottocartelle* (che si occupa sempre di trovare tutti i file pertinenti all'interno delle sottocartelle della directory).

Viene caricato il file CSV completo specificato dal path di input in un DataFrame e vengono selezionate le colonne numeriche a partire dalla seconda colonna. Trova il valore minimo e il valore massimo tra tutti i dati delle colonne selezionate e normalizza tutte le colonne usando la formula (valore - min_val) / (max_val - min_val), che scala i dati in modo che siano compresi tra 0 e 1.
Si concatena la prima colonna del DataFrame originale con i dati normalizzati e viene salvato il nuovo Dataframe.

```
def normalizza_e_concatena(input_csv_path,
↪  output_csv_path):
    df = pd.read_csv(input_csv_path, sep=';')

    colonne_numeriche = df.iloc[:,1:].
    select_dtypes(include=[np.number])

    min_val = colonne_numeriche.min().min()
    max_val = colonne_numeriche.max().max()

    normalized_columns = (colonne_numeriche -
    ↪  min_val) / (max_val - min_val)

    df_original = pd.read_csv(input_csv_path,
    ↪  sep=';')

    df_out = pd.concat([df_original.iloc[:, :1],
    ↪  normalized_columns], axis=1)

    df_out.to_csv(output_csv_path, index=False,
    ↪  sep=';')

    print(f'Dati normalizzati e concatenati salvati
    ↪  in: {output_csv_path}')
```

In particolar modo questo script viene utilizzato sui file "data_cna.csv", "data_mrna_seq_v2_rsem.csv" e "data_rppa.csv".

### 5.4.6 Cambio Header delle label

Considerato che in alcuni file "data_methylation.csv" la prima label non segue uno schema uniforme, ma contiene lo stesso dato, è stato sviluppato uno script in grado di standardizzare l'header della prima colonna.
Il ciclo for viene utilizzato per scorrere tutti i file e le sottocartelle all'interno della cartella principale. Per ogni file trovato, si verifica se il nome del file è "data_methylation_hm450.csv" e in caso affermativo viene composto il path completo del file.
Il file CSV viene aperto in modalità lettura, vengono lette tutte le righe e vengono memorizzate in una lista.

```
def cambia_label_header(path_cartella_principale,
↪  nuova_label):
    for cartella, sottocartelle, file in
    ↪  os.walk(path_cartella_principale):
        for nome_file in file:
            if nome_file ==
            ↪  'data_methylation_hm450.csv':
                percorso_file =
                ↪  os.path.join(cartella,
                ↪  nome_file)

                with open(percorso_file, 'r',
                ↪  newline='') as file_csv:
                    reader = csv.reader(file_csv,
                    ↪  delimiter=';')
                    righe = list(reader)
```

Se la prima riga non inizia con la nuova label, la modifica. Viene aperto il file CSV in modalità scrittura e vengono scritte tutte le righe modificate nel file.

```
if len(righe) > 0 and len(righe[0]) > 0:
    if righe[0][0] != nuova_label:
        righe[0][0] = nuova_label

        with open(percorso_file, 'w', newline='')
        ↪  as file_csv_modificato:
            writer =
            ↪  csv.writer(file_csv_modificato,
            ↪  delimiter=';')
            writer.writerows(righe)

        print(f"Label cambiata nel file:
        ↪  {percorso_file}")

cartella_principale = '/Dataset'
nuova_label = 'Hugo_Symbol'
cambia_label_header(cartella_principale,
↪  nuova_label)
```

### 5.4.7 Split Methylation

La maggior parte dei progetti include il file relativo alla metilazione con l'etichetta 'hm450', ma una piccola parte di essi contiene altri due tipi di file: la metilazione con l'etichetta 'hm27' o la metilazione combinata 'hm27_hm450'.
Le etichette si riferiscono a due diverse versioni delle piattaforme di analisi della metilazione del DNA:

- 'hm27' analizza circa 27.000 siti di metilazione nel genoma umano e fornisce una copertura limitata di siti di metilazione
- 'hm450' analizza circa 450.000 siti di metilazione nel genoma umano e offre una copertura molto più completa del genoma rispetto a hm27, permettendo una visione più dettagliata della metilazione del DNA

Alla luce di ciò è stato scelto di utilizzare il file con l'etichetta 'hm450', in quanto offre una visione molto più completa del profilo di metilazione del DNA rispetto all'hm27 e inoltre include i siti di 'hm27'.
Dato che i file di metilazione combinata 'hm27_hm450' hanno in alcune righe due geni nella stessa cella è stato implementato uno script Python *split_methylation* per separare questi valori.

Lo script contiene una funzione *separa_duplica_valori* che carica un file CSV, separa i valori nella colonna 'Hugo_Symbol' che sono separati da virgole, duplica le righe in base a questi valori separati e salva il DataFrame modificato.

```
def separa_duplica_valori(file_path):
    df = pd.read_csv(file_path, delimiter=';')

    df = df.dropna(subset=['Hugo_Symbol'])

    df['Hugo_Symbol'] =
    ↪  df['Hugo_Symbol'].str.split(',')

    df = df.explode('Hugo_Symbol')

    df.to_csv(file_path, index=False, sep=';')
```

## 5.5 Data Merge

Dopo aver lavorato sui singoli file di ciascun progetto è stata avviata una fase di unione per comporre il Dataset completo, comprensivo di soli 6 file.

### 5.5.1 Patient

Si inizializza una lista vuota che verrà utilizzata per salvare i frame dei singoli file CSV e vengono attraversate tutte le sottocartelle. Per ogni file all'interno delle sottocartelle, verifica se il nome del file termina con ".csv" e inizia con "data_clinical_patient". Se il file soddisfa i criteri, legge il CSV e aggiunge il frame risultante alla lista. Dopo aver attraversato tutti i file, concatena tutti i frame in uno unico.

```
def concat_and_remove_duplicates(main_folder):
    frames = []

    for subdir, _, files in os.walk(main_folder):
        for file in files:
            if file.endswith(".csv") and
            ↪  file.startswith
                ("data_clinical_patient"):
                file_path = os.path.join(subdir,
                ↪  file)
                df = pd.read_csv(file_path,
                ↪  delimiter=';')
                frames.append(df)

    concatenated_df = pd.concat(frames,
    ↪  ignore_index=True)
```

Le celle vuote del frame risultante vengono riempite con la stringa '[Not Available]'. Dopodichè vengono eliminati i duplicati basandosi sulla colonna 'SAMPLE_ID' e viene salvato il frame risultante in un nuovo file CSV.

```
for column in concatenated_df.columns:
    concatenated_df[column].fillna('[Not
    ↪  Available]', inplace=True)

deduplicated_df = concatenated_df.drop_duplicates
                  (subset='SAMPLE_ID')

deduplicated_df.to_csv(
    '/Dataset/data_clinical_patient.csv',
    index=False, sep=';')

concat_and_remove_duplicates(main_folder)
```

### 5.5.2 Mutation

Viene inizializzata una lista vuota, che verrà utilizzata per memorizzare i dati dei file CSV letti, e avviato un ciclo for per esplorare i file e le cartelle all'interno della directory. Viene verificato se il nome del file finisce con "data_mutations.csv" e in caso positivo si legge il file CSV, si memorizza in un DataFrame e viene aggiunto alla lista.

```
def leggi_csv(cartella):
    dati = []

    for root, dirs, files in os.walk(cartella):
        for file in files:
            if file.endswith("data_mutations.csv"):
                percorso_file = os.path.join(root,
                ↪  file)
                df = pd.read_csv(percorso_file,
                ↪  delimiter=';')
                dati.append(df)
    return dati
```

Dopodichè vengono concatenati i risultati in un unico DataFrame e ne viene creato uno nuovo con le righe duplicate rimosse.

```
cartella_principale = "/Dataset (completo)"
dati_concatenati =
↪  pd.concat(leggi_csv(cartella_principale),
↪  ignore_index=True)

dati_senza_duplicati =
↪  dati_concatenati.drop_duplicates()

percorso_output = "/Dataset/data_mutations.csv"
dati_senza_duplicati.to_csv(percorso_output,
↪  index=False, sep=';')
```

### 5.5.3 CNA/mRNA/RPPA

Dato che la struttura dei file "data_cna.csv", "data_mrna_seq_v2_rsem.csv" e "data_rppa.csv" è molto simile viene utilizzato qusesto script per tutti.

Viene letto il CSV di input e viene memorizzato in un DataFrame. Si ottiene il path della directory specificata e viene creata una lista di tutte le sottodirectory nella directory principale.
Dopodichè viene creata una copia del DataFrame di input, che sarà utilizzata per concatenare i dati.

```
def concatena_csv(input_csv, output_csv,
↪  directory_path):
    input_df = pd.read_csv(input_csv,
    ↪  delimiter=';')

    main_directory =
    ↪  os.path.abspath(directory_path)

    subdirectories = [dir for dir in
    ↪  os.listdir(main_directory) if
    ↪  os.path.isdir(os.path.join(main_directory,
    ↪  dir))]

    merged_df = input_df.copy()
```

Successivamente, il codice itera su ciascuna sottodirectory trovata. Ottiene il percorso completo del file 'data_cna.csv', legge il suo contenuto in un DataFrame e lo concatena. Infine rimuove i duplicati basandosi sulla colonna "Hugo_Symbol" e sulle colonne successive.

```
for subdirectory in subdirectories:
    subdirectory_path =
    ↪  os.path.join(main_directory, subdirectory)

    sub_file_path = os.path.join(subdirectory_path,
    ↪  'data_cna.csv')

    if os.path.exists(sub_file_path) and
    ↪  os.path.isfile(sub_file_path):
        sub_df = pd.read_csv(sub_file_path,
        ↪  delimiter=';')

        merged_df = pd.concat([merged_df, sub_df])

        merged_df = merged_df.drop_duplicates
                    (subset=["Hugo_Symbol"] +
                    ↪  list(merged_df.columns[1:]))
```

Dopodichè si ordina il DataFrame in base alla colonna "Hugo_Symbol" e si esegue una fusione delle righe con lo stesso valore nella colonna "Hugo_Symbol", mantenendo solo la prima occorrenza di ciascun valore. Viene salvato il Dataframe finale come un nuovo file CSV.

```
    merged_df =
    ↪  merged_df.sort_values(by="Hugo_Symbol")

    merged_df = merged_df.groupby('Hugo_Symbol',
    ↪  as_index=False).first().fillna(method='ffill')

    merged_df.to_csv(output_csv, index=False,
    ↪  sep=';')
```

Esempio di utilizzo:

```
input_csv = '/acc_tcga/data_cna.csv'
output_csv = '/Dataset/data_cna.csv'
directory_path = 'Scrivania/data'
concatena_csv(input_csv, output_csv,
↪  directory_path)
```

# 6 Modello

In questa sezione verrà descritto l'intero workflow relativo alla fase sperimentale, comprensivo di una descrizione dettagliata di ciascun passaggio e delle relative metodologie adottate.

## 6.1 Fasi preliminari

### 6.1.1 Scelta dei geni

Il file data_mrna_seq_v2_rsem ha gli header delle colonne corrispondenti ai geni e le celle contengono i valori di espressione genica per ciascun gene in diversi pazienti. Per ciascuna colonna (gene), viene calcolata la deviazione standard dei valori di espressione genica.
Dopodichè vengono filtrati i geni con deviazione standard superiore ad una certa soglia, i geni che superano questa soglia saranno quelli che mostrano una maggiore variabilità nei livelli di espressione tra i pazienti nel dataset.
Sono state "costruite" diverse varianti di questo file in base a diverse soglie:

- Soglia a 0.00005, a cui corrispondono 12670 geni
- Soglia a 0.0001, a cui corrispondono 7670 geni
- Soglia a 0.0002, a cui corrispondono 3748 geni
- Soglia a 0.0005, a cui corrispondono 1349 geni
- Soglia a 0.0010, a cui corrispondono 610 geni
- Soglia a 0.0015, a cui corrispondono 399 geni
- Soglia a 0.0020, a cui corrispondono 275 geni
- Soglia a 0.0025, a cui corrispondono 216 geni
- Soglia a 0.0030, a cui corrispondono 166 geni

Le features derivanti dalle varianti di questo file sono state integrate nel dataset utilizzato per l'addestramento del modello. Nel caso in cui si utilizzi il file completo si avrebbe un numero di geni pari a 23665.

### 6.1.2 Dataset per la classificazione

Il dataset utilizzato per l'addestramento del modello è composto da un insieme di feature di tipo clinico e dalla sequenza di geni filtrata mediante la deviazione standard. Di questo dataset sono state costruite tre varianti:

- **Only Mutations**: le cui features sono esclusivamente le espressioni geniche per ciascun paziente
- **Only Variants**: le cui features sono date solamente dal conteggio del tipo di mutazioni per ogni gene (divise tra SNP, DEL, INS) e CNA
- **Mutations and Variants**: le cui features includono sia le espressioni geniche che il conteggio del tipo di mutazioni

I tipi di alterazioni genetiche (`Variants_Type`) sono state tenute in cosiderazione perchè possono giocare un ruolo significativo nello sviluppo dei tumori, poiché possono influenzare la regolazione genica, la funzione delle proteine e altri processi cellulari cruciali. L'identificazione di queste mutazioni è importante per la comprensione dei meccanismi sottostanti al cancro e per lo sviluppo di approcci terapeutici mirati come analizzato nel seguente lavoro.

## 6.2 Classificazione

Il modello è progettato per la classificazione del tipo di cancro. Per fare ciò, utilizza uno dei tre dataset menzionati in precedenza come input.
L'obiettivo finale è determinare la tipologia specifica di cancro associata ai dati forniti, basandosi sulle feature presenti nel dataset selezionato.

### 6.2.1 Funzioni di utilità

Vengono utilizzate un insieme di funzioni di utilità per la classificazione:

- *pre_processing*, si occupa della la preparazione dei dati, inclusa la codifica one-hot delle variabili categoriche
- *weighted_categorical_crossentropy*, definisce una funzione di perdita pesata
- *created_model*, definisce l'architettura del modello per la classificazione
- *classification_model*, si occupa di creare, addestrare e valutare il modello di classificazione
- *eval_dnn*, valuta le prestazioni della rete neurale

### 6.2.2 Costruzione del modello

La struttura del modello è definita nella funzione *created_model* e comprende:

- **Input Layer**
- **Strati di Convoluzione**, utilizzati per estrarre features specifiche mediante filtri con dimensioni diversificate e funzione di attivazione ReLU
- **Strati di Pooling**, utilizzati per ridurre la dimensione spaziale dell'output dalla convoluzione, preservando le feature più significative
- **Flatten**, trasforma l'output tridimensionale dei layer precedenti in un vettore monodimensionale
- **Output Layer**, produce l'output finale del modello, con un numero di unità pari al numero di classi e attivazione softmax per ottenere le probabilità di appartenenza a ciascuna classe

```
Model: "classification"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 332, 1)]          0

 conv1d (Conv1D)             (None, 7, 256)            13056

 conv1d_1 (Conv1D)           (None, 7, 128)            327808

 max_pooling1d (MaxPooling1  (None, 3, 128)            0
 D)

 conv1d_2 (Conv1D)           (None, 3, 128)            82048

 max_pooling1d_1 (MaxPoolin  (None, 1, 128)            0
 g1D)

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 26)                3354

=================================================================
Total params: 426266 (1.63 MB)
Trainable params: 426266 (1.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 12: Model Summary Classificazione

#### 6.2.3 Pesi classificazione

Per quanto riguarda la questione dei pesi per le classi sbilanciate, il codice implementa la funzione weighted_categorical_crossentropy che introduce una ponderazione delle classi in base alla distribuzione dei sample. Questo approccio assicura che il modello attribuisca un peso maggiore alle classi sottorappresentate. La scelta dei pesi è basata sul rapporto tra il numero totale di sample nel dataset e la quantità di sample presenti per ciascuna classe. Questo permette al modello di prestare maggiore attenzione alle classi meno comuni, migliorando così la capacità del modello di generalizzare.

### 6.3 Rete Siamese

#### 6.3.1 Funzioni di utilità

Vengono utilizzate un insieme di funzioni di utilità per il corretto funzionamento della rete:

- *initialize_bias*, inizializza il bias dei layer con valori normalmente distribuiti
- *last_layer*, crea l'ultimo strato della rete siamese, che può essere L1, L2 o una funzione di somiglianza coseno (nel nostro caso abbiamo usato L2)
- *make_oneshot_task*, seleziona un set di supporto di N campioni da un set di test e un campione di test
- *indices_save*, crea un dizionario dove le chiavi sono i tipi di cancro e i valori sono gli indici delle righe corrispondenti nel dataset
- *siamese_network*, prepara il modello siamese per l'addestramento, includendo la compilazione del modello e la preparazione del dataset

#### 6.3.2 get_siamese_model

La funzione *get_siamese_model* è responsabile della creazione del modello della rete siamese. Il modello è composto da due reti neurali identiche, ciascuna delle quali elabora uno dei due input (coppia di campioni da confrontare).
Vengono definiti due input chiamati left_input e right_input, ciascuno con la forma input_shape.
La rete siamese è implementata come un modello sequenziale chiamato siamese_model. Gli output delle due reti siamesi vengono calcolati separatemente per left_input e right_input e successivamente passati attraverso la funzione last_layer per calcolare la distanza o somiglianza tra le rappresentazioni.

#### 6.3.3 get_batch

La funzione get_batch è utilizzata per creare un batch di coppie di esempi per l'addestramento della rete.
La funzione genera due array vuoti pairs e targets, dove pairs conterrà le coppie di dati e targets le etichette (1 per coppie della stessa classe, 0 per coppie di classi diverse).
La metà del batch (batch_size // 2) è riservata per coppie della stessa classe, mentre l'altra metà è per coppie di classi diverse.
Per ogni coppia, viene scelta casualmente una classe dal dataset di addestramento e

- Per **coppie della stessa classe**, viene selezionato casualmente un campione da quella classe
- Per **coppie di classi diverse**, viene selezionata casualmente una seconda classe diversa dalla prima, e viene scelto un campione da quella classe

Le coppie e le relative etichette vengono aggiunte agli array pairs e targets che vengono restituiti e rappresentano il batch di dati di addestramento.

#### 6.3.4 test_oneshot

La funzione test_oneshot è utilizzata per valutare l'accuratezza del modello su compiti di apprendimento one-shot. La funzione esegue K compiti one-shot.
Per ciascun compito, utilizza la funzione make_oneshot_task per creare coppie di test e supporto. Il modello predice quale coppia del test dovrebbe appartenere alla stessa classe del supporto e l'accuratezza viene calcolata come la percentuale di compiti one-shot correttamente classificati.

### 6.3.5 Struttura della Rete Siamese

La rete siamese è composta da due reti identiche che condividono gli stessi pesi.
Ogni rete è un modello sequenziale con vari layer convoluzionali (Conv1D) e di pooling (MaxPooling1D). Dopo i layer convoluzionali e di pooling, i dati vengono appiattiti (Flatten) e passati all'ultimo layer che calcola la distanza tra le coppie di input.

```
Model: "siamese-network"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_2 (InputLayer)        [(None, 166, 1)]          0         []

 input_3 (InputLayer)        [(None, 166, 1)]          0         []

 sequential (Sequential)     (None, 128)               422912    ['input_2[0][0]',
                                                                  'input_3[0][0]']

 lambda (Lambda)             (None, 128)               0         ['sequential[0][0]',
                                                                  'sequential[1][0]']

 dense_1 (Dense)             (None, 512)               66048     ['lambda[0][0]']

 dropout (Dropout)           (None, 512)               0         ['dense_1[0][0]']

 dense_2 (Dense)             (None, 1)                 513       ['dropout[0][0]']

==================================================================================================
Total params: 489473 (1.87 MB)
Trainable params: 489473 (1.87 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Figure 14: Model Summary Rete Siamese

### 6.3.6 Processo di Addestramento e Valutazione

Il modello viene addestrato utilizzando batch di coppie di esempi. Periodicamente, il modello viene valutato su compiti di apprendimento one-shot per misurare la sua accuracy.
Il processo continua per un numero prefissato di iterazioni e alla fine viene confrontato il miglior modello. Per quanto riguarda l'analisi dei dataset che contengono sia espressione genomica che l'informazione riguardante le varianti, è stato utilizzato lo stesso modello con l'aggiunta di due layer sia in fase di classificazione sia durante l'apprendimento della rete siamese.

## 7 Analisi dei risultati

Sono state eseguite diverse sperimentazioni impiegando principalmente due dataset menzionati in precedenza, ovvero quello generato con una soglia di 0.0030 e quello generato con una soglia di 0.0005.
Questi esperimenti possono essere suddivisi in due approcci distinti:

- Nella prima modalità, è stata utilizzata la medesima architettura di rete proposta nel paper CancerSiamese, conducendo diversi test esclusivamente basati sul dataset **Only Mutations** ed altri con il dataset **Mutations and Variants**
- Nella seconda modalità è stata utilizzata una rete più profonda, caratterizzata dall'aggiunta di due layer supplementari, ed è stata eseguita unicamente sul dataset **Mutations and Variants**

Dalle sperimentazioni effettuate si può notare che un incremento della batch size porta ad un miglioramento dell'accuracy in quanto consente di elaborare simultaneamente un maggior numero di blocchi di dati durante la fase di addestramento.

| Dataset | Tipo di Dataset | batch_size | #Classi | #Pazienti | #Features | evalaute_every | n_iterazioni | n_val | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 0.0030 | Only Mutations | 26 | 26 | 9757 | 166 | 200 | 100000 | 1000 | 86.1 % |
| | | 128 | | | | | | | 86.6 % |
| | | 256 | | | | | | | 89.8 % |
| | | 512 | | | | | | | 91.1 % |
| 0.0005 | | 26 | | | 1349 | | | | 91.1 % |

(a) Dataset Only Mutations con struttura di CancerSiamese

| Dataset | Tipo di Dataset | batch_size | #Classi | #Pazienti | #Features | evalaute_every | n_iterazioni | n_val | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 0.0030 | Mutations and Variants SNP DEL INS + CNA | 26 | 23 | 5394 | 691 | 200 | 100000 | 1000 | 36.7 % |
| | | 128 | | | | | | | 36.4 % |
| | | 256 | | | | | | | 32.7 % |
| | | 512 | | | | | | | 29.7 % |
| 0.0005 | | 26 | 25 | 8796 | 5719 | | | | 22.4 % |

(b) Dataset Mutations and Variants con struttura di CancerSiamese

| Dataset | Tipo di Dataset | batch_size | #Classi | #Pazienti | #Features | evalaute_every | n_iterazioni | n_val | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 0.0030 | Mutations and Variants SNP DEL INS + CNA | 26 | 23 | 5394 | 691 | 200 | 100000 | 1000 | 20.8 % |
| | | 128 | | | | | | | 74.5 % |
| | | 256 | | | | | | | 75.4 % |
| 0.0005 | | 26 | 25 | 8796 | 5719 | | | | 13.5 % |
| | | 32 | | | | | | | 85.3 % |

(c) Dataset Mutations and Variants con struttura della rete più profonda

Figure 13: Tabella riepilogativa delle sperimentazioni

## 8 Conclusioni

Questo lavoro ha dimostrato la possibilità di prevedere tipi di cancro anche quando si dispone di un numero limitato di campioni. Ciò potrebbe ispirare nuove e innovative applicazioni di soluzioni di apprendimento one-shot e few-shot per migliorare la diagnosi del cancro, la prognosi e la nostra comprensione dei meccanismi alla base del cancro.

## 9 Data Availability

I dataset TCGA utilizzati per la costruzione del Dataset principale, utilizzato in questo lavoro, sono reperibili su cBioPortal for Cancer Genomics.
Il dataset principale e le varianti del file utilizzato per la fase di testing sono disponibili su Google Drive.
Il codice sviluppato è disponibile su GitHub.

## References


[1] *Artificial Intelligence, Machine Learning and Genomics*

[2] *The Cancer Genome Atlas Program (TCGA)*

[3] *Sebastian, A.M.; Peter, D.*
*Artificial Intelligence in Cancer Research: Trends, Challenges and Future Directions. Life 2022, 12, 1991.*

[4] *Jiao, W., Atwal, G., Polak, P. et al.*
*A deep learning system accurately classifies primary and metastatic cancers using passenger mutation patterns. Nat Commun 11, 728 (2020)*

[5] *Chiu, Y. C., Zheng, S., Wang, L. J., Iskra, B. S., Rao, M. K., Houghton, P. J., Huang, Y., & Chen, Y. (2021)*
*Predicting and characterizing a cancer dependency map of tumors with deep learning. Science advances, 7(34), eabh1275*

[6] *Mostavi, M., Chiu, YC., Chen, Y. et al.*
*CancerSiamese: one-shot learning for predicting primary and metastatic tumor types unseen during model training. BMC Bioinformatics 22, 244 (2021)*

[7] *Chicco D.*
*Siamese Neural Networks: An Overview. Methods in molecular biology (Clifton, N.J.), 2190, 73–94.*

[8] *Cerami et al.*
*The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data. Cancer Discovery. May 2012 2; 401.*

[9] *Liu, Y., Qu, HQ., Mentch, F.D. et al.*
Application of deep learning algorithm on whole genome sequencing data uncovers structural variants associated with multiple mental disorders in African American patients. Mol Psychiatry 27, 1469–1478 (2022)

[10] *HUGO Gene Nomenclature Committee*