



**UNIVERSITÀ DEGLI STUDI DI CATANIA**  
DIPARTIMENTO DI MATEMATICA E INFORMATICA  
CORSO DI LAUREA MAGISTRALE IN INFORMATICA

---

*Giuseppe Condorelli*

Denoising Autoencoder

---

PROGETTO DI FINE CORSO

---

Prof. Giovanni Maria Farinella

---

Anno Accademico 2023 - 2024

# Contents

<b>1 Problema</b>	<b>1</b>
1.1 Introduzione . . . . .	1
1.1.1 Reperibilità del materiale . . . . .	2
<b>2 Dataset</b>	<b>3</b>
2.1 Fase di acquisizione dei dati . . . . .	3
2.2 Operazioni preliminari sui dati . . . . .	4
2.3 Operazioni sul dataset . . . . .	5
2.3.1 Cropping e Resize . . . . .	5
2.3.2 Aggiunta del rumore . . . . .	6
2.4 Data augmentation . . . . .	7
2.5 Dati di training, dati di validation e dati di test . . . . .	8
2.6 Organizzazione cartelle del dataset . . . . .	8
<b>3 Metodi</b>	<b>9</b>
3.1 DataLoader . . . . .	9
3.2 Seed . . . . .	9
3.3 Autoencoder . . . . .	9
3.4 Training . . . . .	10
<b>4 Valutazione</b>	<b>12</b>
4.1 Metriche e grafici . . . . .	12
<b>5 Esperimenti</b>	<b>13</b>
5.1 Primo esperimento . . . . .	13
5.2 Secondo esperimento . . . . .	17
5.3 Terzo esperimento . . . . .	21
5.4 Quarto esperimento . . . . .	25

5.5	Quinto esperimento . . . . .	29
5.6	Sesto esperimento . . . . .	33
5.7	Settimo esperimento . . . . .	37
5.8	Ottavo esperimento . . . . .	41
5.9	Nono esperimento . . . . .	45
5.10	Decimo esperimento . . . . .	49
<b>6</b>	<b>Demo</b>	<b>52</b>
<b>7</b>	<b>Codice</b>	<b>55</b>
7.1	Notebook Denoising_Autoencoder.ipynb . . . . .	55
7.1.1	Preprocessing immagini . . . . .	55
7.1.2	Utility functions . . . . .	56
7.1.3	Esperimenti . . . . .	56
7.1.4	Demo . . . . .	57
<b>8</b>	<b>Conclusioni</b>	<b>58</b>
8.1	Considerazioni riguardanti il dataset . . . . .	58
8.2	Considerazioni riguardanti gli esperimenti . . . . .	58
8.3	Lezioni apprese . . . . .	59
8.4	Sviluppi futuri . . . . .	60
	<b>Bibliography</b>	<b>61</b>

# List of Figures

2.1	Esempio con watermark . . . . .	4
2.2	Esempio di cropping . . . . .	5
2.3	(a) Originale (b) Chromatic Noise (c) Luminance Noise . . . . .	7
5.1	Risultati primo esperimento . . . . .	16
5.2	Risultati secondo esperimento . . . . .	20
5.3	Risultati terzo esperimento . . . . .	24
5.4	Risultati quarto esperimento . . . . .	28
5.5	Risultati quinto esperimento . . . . .	32
5.6	Risultati sesto esperimento . . . . .	36
5.7	Risultati settimo esperimento . . . . .	40
5.8	Risultati ottavo esperimento . . . . .	44
5.9	Risultati nono esperimento . . . . .	48
5.10	Risultati decimo esperimento . . . . .	51
6.1	Immagini demo senza rumore . . . . .	52
6.2	Immagini demo con rumore . . . . .	53
6.3	Immagini demo ricostruite senza rumore . . . . .	54

# List of Tables

2.1	Numero di immagini ottenute pre e dopo l'operazioni preliminari . . . . .	5
5.1	Risultati degli Esperimenti . . . . .	13
5.2	Modello primo esperimento . . . . .	14
5.3	Valori medi di PSNR e SSIM . . . . .	15
5.4	Modello secondo esperimento . . . . .	17
5.5	Valori medi di PSNR e SSIM . . . . .	19
5.6	Modello terzo esperimento . . . . .	21
5.7	Valori medi di PSNR e SSIM . . . . .	23
5.8	Model Architecture . . . . .	25
5.9	Valori medi di PSNR e SSIM . . . . .	27
5.10	Modello quinto esperimento . . . . .	30
5.11	Valori medi di PSNR e SSIM . . . . .	31
5.12	Modello sesto esperimento . . . . .	33
5.13	Valori medi di PSNR e SSIM . . . . .	35
5.14	Modello settimo esperimento . . . . .	37
5.15	Valori medi di PSNR e SSIM . . . . .	39
5.16	Modello ottavo esperimento . . . . .	41
5.17	Valori medi di PSNR e SSIM . . . . .	43
5.18	Modello nono esperimento . . . . .	45
5.19	Valori medi di PSNR e SSIM . . . . .	47
5.20	Modello decimo esperimento . . . . .	49
5.21	Valori medi di PSNR e SSIM . . . . .	50

# Chapter 1

## Problema

### 1.1 Introduzione

Il presente progetto si propone di affrontare la sfida di rimozione del rumore da immagini presenti in fotografia utilizzando una architettura autoencoder con layers di Convolutional Neural Network (CNN), un'architettura particolarmente diffusa ed adatta nel cogliere le proprietà nascoste delle immagini.

L'obiettivo primario è sviluppare un sistema automatico e preciso in grado di identificare e rimuovere il rumore dalle immagini.

La scelta di utilizzare un autoencoder basato su strati di CNN deriva dalla sua natura intrinsecamente adatta al trattamento di immagine. L'architettura presenta una notevole capacità di apprendimento delle caratteristiche intrinseche delle immagini fornite. Questa architettura è stata impiegata in diverse applicazioni con risultati promettenti, dimostrando la sua versatilità e robustezza [1, 2].

Un altro aspetto fondamentale di questo progetto riguarda l'acquisizione e la preparazione del dataset. Il focus del progetto è la rimozione di rumore in scatti wildlife presi dai fotografi. A tal fine le immagini sono state acquisite sfruttando il motore di ricerca Google mediante un plug-in di Google Chrome. Vista la metodologia grezza di acquisizione, il dataset è stato successivamente sottoposto a operazioni di pulizia, includendo il filtraggio delle immagini di scarsa qualità e con watermark visibili, al fine di ottenere un dataset finale ottimizzato e bilanciato. Infine le immagini sono state sottoposte ad introduzione di rumore gaussiano luminoso e cromatico.

Dopo aver stabilito l'architettura di partenza ed ottenuto il dataset la mia attenzione si è spostata sullo sviluppare diversi esperimenti al fine di valutare le prestazioni dei modelli, questo argomento è infatti abbondantemente approfondito nei capitoli 3, 4 e 5. Infine una spiegazione sommaria dei codici ausiliari utilizzati durante il progetto è presente nel capitolo 7.

### 1.1.1 Reperibilità del materiale

Tutto il materiale sviluppato, incluso il dataset è reperibile alla seguente repo [3] di GitHub.

# Chapter 2

## Dataset

In questo capitolo, descriverò in dettaglio il dataset utilizzato per addestrare e valutare la bontà dell'autoencoder coprendo i seguenti punti:

1. Fase di acquisizione
2. Operazioni preliminari (Pulizia ed aggiunta del rumore)
3. Data augmentation
4. Dati di training e dati di test
5. Organizzazione cartelle del dataset

### 2.1 Fase di acquisizione dei dati

Una fase fondamentale per la buona riuscita del progetto è stata quella di trovare il dataset ideale per affrontare il problema.

L'obiettivo del progetto è quello di trattare foto prese da fotografi wildlife, a tal fine ho dunque optato per una semplice ricerca su Google Immagini usando la parola chiave "Wildlife bird shots". Dopodichè ho scaricato l'intero pacco immagini forniti da google sfruttando uno dei tanti plugin disponibili su Google Chrome Extensions.

La scelta di questa tipologia "grezza" di acquisizione dati mi ha permesso di ottenere un'ampia varietà di foto uniche e diverse tra loro utile per costruire un semplice dataset adatto per sfruttare le nuove conoscenze acquisite durante il corso

ed in linea con capacità hardware a me disponibili.

La fase di acquisizione mi ha permesso di ottenere un totale di 876 immagini.

## 2.2 Operazioni preliminari sui dati

Dopo aver completato il processo di acquisizione, mi sono dedicato a garantire la qualità delle immagini raccolte, operazione particolarmente necessaria giustificata dalla metodologia di acquisizione scelta.



Figure 2.1: Esempio con watermark

Un'attenzione particolare è stata data alla rimozione di immagini di scarsa qualità o contenenti elementi non coerenti come watermark (Fig.2.1) e, in presenza di immagini con cornici, al cropping out di quest'ultime (Fig. 2.2).

Questo mi ha permesso di ottenere un dataset contenente solo immagini nitide e coerenti col mio goal.

Le operazioni effettuate sul dataset hanno portato il dataset finale ad avere un totale di 501 immagini. (Tab.2.1)

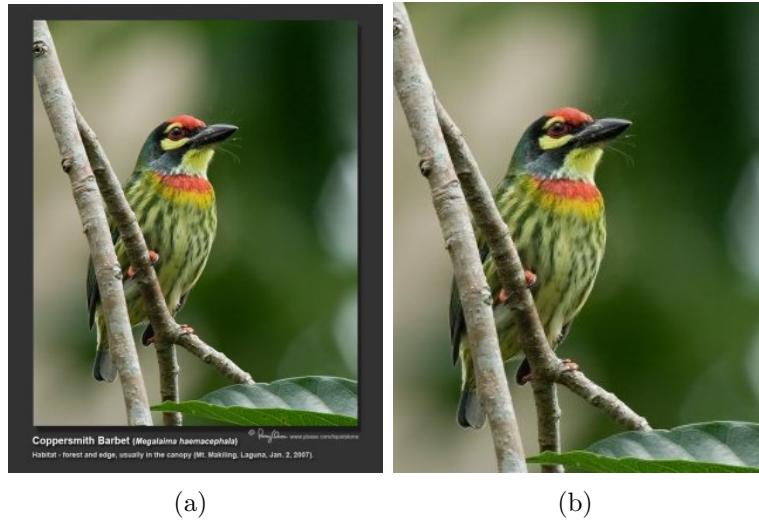


Figure 2.2: Esempio di cropping

Stato	Numero di immagini
Pre-pulizia	876
Post-pulizia	501

Table 2.1: Numero di immagini ottenute pre e dopo l'operazioni preliminari

## 2.3 Operazioni sul dataset

### 2.3.1 Cropping e Resize

Oltre alle operazioni preliminari spiegate precedentemente, ho apportato ulteriori modifiche al dataset al fine di renderlo più adatto per il training dell'autoencoder.

L'obiettivo principale in questa fase è stato quello di standardizzare le dimensioni delle immagini contenute nel dataset in dimensioni fisse di 128x128 al fine di poterle usare con il modello.

Tuttavia, al fine di mantenere intatte le informazioni strutturali delle immagini, un semplice resizing sarebbe andato in contro ai miei obiettivi.

Infatti il dataset presentava immagini di varie dimensioni spesso con lati diversi

tra loro, portando l'immagine a presentarsi con una forma rettangolare e, un semplice resizing, le avrebbe potute distorcere ed appiattire, modificando l'aspetto visivo del dato.

Per ovviare a questo problema ho scelto di adottare un cropping centrale in tutte le immagini, prendendo, come dimensione di riferimento, la lunghezza del lato minimo. Questa soluzione ha cancellato alcune informazioni dalle foto che, essendo non particolarmente rilevanti per fare denoising, portano ad un buon compromesso per poterne mantenere le strutture.

Una volta effettuata la fase di cropping ho dunque continuato eseguendo un resize delle immagini a 128x128.

### 2.3.2 Aggiunta del rumore

Altro aspetto importante per la buon riuscita del training dell'architettura è stato esaminare le due tipologie di rumore più frequenti in fotografia ed applicarle al dataset.

In fotografia è infatti possibile distinguere due tipologie di rumore: cromatico e luminoso.

Il rumore cromatico è caratterizzato dalla presenza di rumore nei canali di colore ed è distinguibile dalla caratteristica di presentare, nelle immagini affette, delle chiazzette colorate.

Il rumore luminoso, al contrario, presenta rumore nel canale della luminanza e si presenta con puntini neri e bianchi.

Individuate le due tipologie di rumore, li ho dunque aggiunti al mia dataset sfruttando una gaussiana. (Fig.2.3)

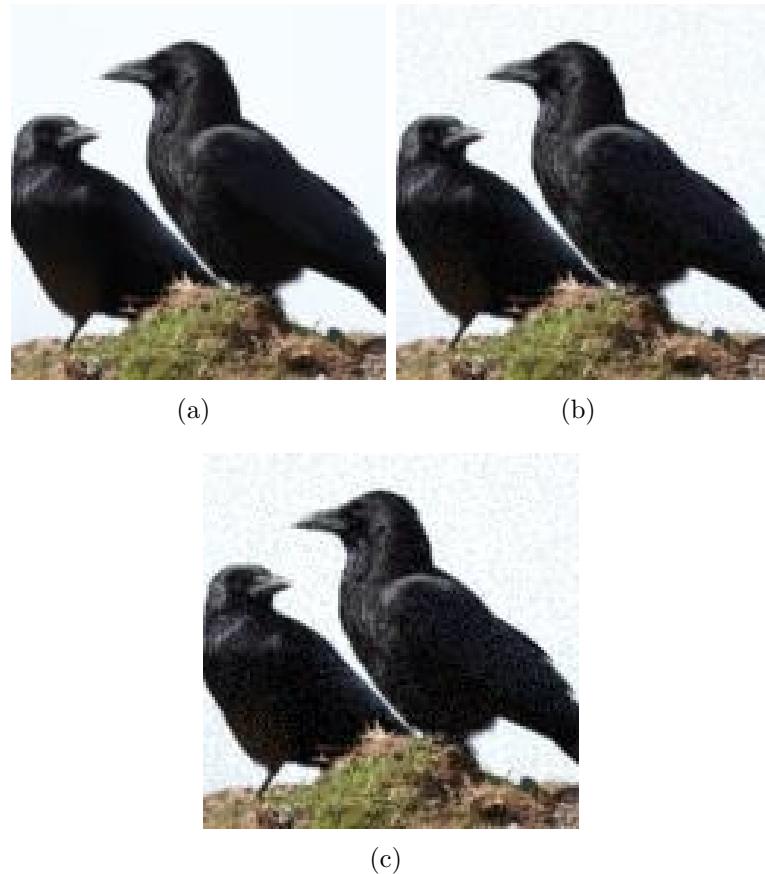


Figure 2.3: (a) Originale (b) Chromatic Noise (c) Luminance Noise

## 2.4 Data augmentation

Vista la scarsità di immagini presenti nel dataset, ho deciso di applicare il rumore cromatico e luminoso alle stesse immagini e di unire i 2 nuovi dataset rumorosi ottenuti in uno.

Questo mi ha permesso di raddoppiarne le dimensioni permettendo al modello di poter apprendere da più esempi.

## 2.5 Dati di training, dati di validation e dati di test

Il dataset non è stato diviso in cartelle separate di training e validation.

Per il suo split ho usato una funzione offerta dalla libreria torch chiamata random\_split che permette di dividere il dataset in training e validation.

Il dataset viene diviso in 80% training e 20% validation.

## 2.6 Organizzazione cartelle del dataset

La struttura del dataset è la seguente:

- **dataset:** contiene le immagini senza rumore, utilizzate per il training e validation del modello, usata come ground\_truth per il calcolo della loss.
- **dataset\_chromatic\_noise:** contiene le immagini con rumore cromatico (rumore gaussiano presente in tutti i canali di colore)
- **dataset\_luminance\_noise:** contiene le immagini con rumore luminoso (rumore gaussiano presente solo nel canale luminoso).

# Chapter 3

## Metodi

### 3.1 DataLoader

Per caricare il set di immagini su python al fine di essere utilizzati dall'architettura ho utilizzato la funzione DataLoader fornita dalla libreria torch.

La libreria permette di dividere il dataset in batch che vengono poi utilizzare dagli strati implementati nell'architettura.

In particolare, ho impostato il parametro batch\_size a 64 rendendo possibile una fluida esecuzione anche con scarsità di memoria nella mia macchina.

### 3.2 Seed

Al fine di rendere gli esperimenti ripetibili ho utilizzato un seed nelle funzioni randomiche utilizzare, in particolar modo nella fase di splitting del dataset in training e validation.

### 3.3 Autoencoder

Per affrontare il problema proposto, ho utilizzato l'architettura Convolutional Autoencoder, che, data la sua natura, ho ritenuto più opportuna.

Al fine di migliorare le prestazioni dell'architettura, ho adottato una strategia bottom-up, partendo da un'architettura molto semplice contenente pochi layer per poi progressivamente aumentarne la complessità. Questa approccio mi ha permesso

di valutare varie strategie e tecniche sulla base dei risultati per scegliere quella più vantaggiosa.

Ogni architettura proposta presenta diversi strati convoluzionali seguiti spesso da procedure di down-sample e up-sample, quali AvgPool2d, MaxPool2d e ConvTranspose2d, che hanno permesso di realizzare la struttura di encoding e decoding tipica degli autoencoder.

Altri layers importanti sono anche rappresentati dall'utilizzo di funzioni di attivazioni SiLu o LeakyReLU, nonchè layer di BatchNormalization, utili per permettere all'archittetura di cogliere le caratteristiche dei dati non entrando in overfitting.

Per l'apprendimento del modello, ho utilizzato l'algoritmo di ottimizzazione AdamW, che si basa sulla discesa del gradiente, insieme alla funzione MSE loss, particolarmente adatta per training supervisionato.

## 3.4 Training

Per la fase di training ho optato nell'utilizzo di un EarlyStopping, un oggetto fornito da pytorch lightning che ho trovato molto utile.

Quest'ultimo infatti monitora il miglioramento di un valore durante la fase di training e controlla se quest'ultimo migliora nel corso delle epoche.

Ho usato questo oggetto per controllare che la "val loss" migliorasse di almeno 0.005 entro 10 epoche, se così non fosse, quello che accadrebbe è che la fase di training verrebbe interrotta dall'oggetto segnalando al trainer di fermarla.

Ho deciso di optare per questa soluzione per fermare in automatico il training nel momento in cui l'architettura smettesse di convergere così da ottimizzare l'uso delle risorse e del tempo.

E' importante inoltre specificare che il max\_epoch del trainer utilizzato è 50.

# Chapter 4

## Valutazione

### 4.1 Metriche e grafici

Le metriche e grafici utilizzati sono:

- **Training and Validation Loss:** Questo grafico permette di capire se il modello sta riuscendo a convergere col passare delle epoche, dimostrando dunque la sua capacità nell'apprendere e validità.
- **PSNR:** Questa metrica permette di, resa disponibile l'immagine originale, di valutare il grado di rumore in dB presente nell'immagine di confronto, particolarmente utile per controllare il livello di rumore nell'immagine rumorosa in input e in quella ricostruita dal modello. Più è basso è il valore meno rumore è presente.
- **SSIM:** Una metrica molto utile per valutare il livello di somiglianza tra due immagini e dunque valutare la bontà del risultato proposto dal modello rispetto all'immagine originale senza rumore.

Durante gli esperimenti, osserveremo attentamente tutti i grafici e le metriche citate al fine di valutarne correttamente le capacità e dunque adottare strategie migliori per il miglioramento del modello.

# Chapter 5

## Esperimenti

Table 5.1: Risultati degli Esperimenti

N° Esperimento	PSNR	SSIM	Note
1	11.60 dB	0.2668	
2	12.06 dB	0.2942	Range resample [8x8,128x128]
3	11.45 dB	0.2590	Aumentati i layers di CNN e SiLU
4	16.55 dB	0.3638	Sostituiti Avg con Max, SiLU con LeakyReLU, aggiunti BathNorm e tahn
5	16.41 dB	0.3443	Aggiunti layers di Conv, LeakyReLU e Batch-Norm
6	15.11 dB	0.3251	Aggiunti ulteriori layers
7	17.72 dB	0.4765	Aggiunti altri layers riducendo il resample
8	17.83 dB	0.4713	Cambiamento disposizione layers
9	17.78 dB	0.4483	Kernel size modificata nei primi ed ultimi layers
10	16.85 dB	0.4980	Diminuita complessità

### 5.1 Primo esperimento

Il primo modello risulta essere molto semplice poiché presenta solo 2 fatto di down e up sampling, portando dunque l'immagine a passare da 128x128 a 32x32 in fase di encoding per poi tornare a 128x128 in fase di decoding. Sono anche presenti 6 layers convoluzionali seguiti da funzioni SiLu, utili per cogliere le caratteristiche intrinseche delle immagini nelle varie scale.

Layer (type)	Output Shape	Param #
Conv2d-1	[ -1, 8, 128, 128]	224
SiLU-2	[ -1, 8, 128, 128]	0
AvgPool2d-3	[ -1, 8, 64, 64]	0
Conv2d-4	[ -1, 16, 64, 64]	1,168
SiLU-5	[ -1, 16, 64, 64]	0
AvgPool2d-6	[ -1, 16, 32, 32]	0
Conv2d-7	[ -1, 32, 32, 32]	4,640
SiLU-8	[ -1, 32, 32, 32]	0
ConvTranspose2d-9	[ -1, 16, 64, 64]	4,624
SiLU-10	[ -1, 16, 64, 64]	0
ConvTranspose2d-11	[ -1, 8, 128, 128]	1,160
SiLU-12	[ -1, 8, 128, 128]	0
Conv2d-13	[ -1, 3, 128, 128]	219
<b>Total params:</b>	12,035	
<b>Trainable params:</b>	12,035	
<b>Non-trainable params:</b>	0	

Table 5.2: Modello primo esperimento

Output epochs:

```

Epoch 0: 100%| 13/13 [00:02<00:00, 5.29it/s, v_num=39, train/loss_step=0.335,
    val/loss_step=0.196, val/loss_epoch=0.299, train/loss_epoch=0.299]
Metric val/loss improved. New best score: 0.299
Epoch 2: 100%| 13/13 [00:02<00:00, 5.78it/s, v_num=39, train/loss_step=0.274,
    val/loss_step=0.190, val/loss_epoch=0.291, train/loss_epoch=0.291]
Metric val/loss improved by 0.008 >= min_delta = 0.005. New best score: 0.291
Epoch 4: 100%| 13/13 [00:01<00:00, 6.56it/s, v_num=39, train/loss_step=0.254,
    val/loss_step=0.182, val/loss_epoch=0.281, train/loss_epoch=0.282]
Metric val/loss improved by 0.010 >= min_delta = 0.005. New best score: 0.281
Epoch 5: 100%| 13/13 [00:02<00:00, 6.31it/s, v_num=39, train/loss_step=0.316,
    val/loss_step=0.177, val/loss_epoch=0.274, train/loss_epoch=0.276]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.274
Epoch 6: 100%| 13/13 [00:02<00:00, 6.46it/s, v_num=39, train/loss_step=0.265,
    val/loss_step=0.169, val/loss_epoch=0.264, train/loss_epoch=0.268]
Metric val/loss improved by 0.011 >= min_delta = 0.005. New best score: 0.264
Epoch 7: 100%| 13/13 [00:01<00:00, 6.60it/s, v_num=39, train/loss_step=0.262,
    val/loss_step=0.157, val/loss_epoch=0.246, train/loss_epoch=0.255]
Metric val/loss improved by 0.017 >= min_delta = 0.005. New best score: 0.246
Epoch 8: 100%| 13/13 [00:01<00:00, 6.77it/s, v_num=39, train/loss_step=0.251,
    val/loss_step=0.139, val/loss_epoch=0.219, train/loss_epoch=0.233]
Metric val/loss improved by 0.027 >= min_delta = 0.005. New best score: 0.219
Epoch 9: 100%| 13/13 [00:01<00:00, 6.82it/s, v_num=39, train/loss_step=0.184,
    val/loss_step=0.112, val/loss_epoch=0.179, train/loss_epoch=0.201]
Metric val/loss improved by 0.040 >= min_delta = 0.005. New best score: 0.179
Epoch 10: 100%| 13/13 [00:01<00:00, 6.65it/s, v_num=39, train/loss_step=0.140,
    val/loss_step=0.0804, val/loss_epoch=0.131, train/loss_epoch=0.157]
Metric val/loss improved by 0.049 >= min_delta = 0.005. New best score: 0.131
Epoch 11: 100%| 13/13 [00:02<00:00, 6.41it/s, v_num=39,
    train/loss_step=0.0904, val/loss_step=0.0548, val/loss_epoch=0.0861, train/loss_epoch=0.109]
Metric val/loss improved by 0.044 >= min_delta = 0.005. New best score: 0.086

```

```

Epoch 12: 100%|          | 13/13 [00:01<00:00,  6.76it/s, v_num=39,
    train/loss_step=0.0663, val/loss_step=0.0448, val/loss_epoch=0.0604, train/loss_epoch=0.072]
Metric val/loss improved by 0.026 >= min_delta = 0.005. New best score: 0.060
Epoch 13: 100%|          | 13/13 [00:01<00:00,  6.78it/s, v_num=39,
    train/loss_step=0.0495, val/loss_step=0.045, val/loss_epoch=0.0517, train/loss_epoch=0.0555]
Metric val/loss improved by 0.009 >= min_delta = 0.005. New best score: 0.052
Epoch 15: 100%|          | 13/13 [00:02<00:00,  6.16it/s, v_num=39,
    train/loss_step=0.0455, val/loss_step=0.0405, val/loss_epoch=0.0445, train/loss_epoch=0.0457]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.044
Epoch 20: 100%|          | 13/13 [00:01<00:00,  6.66it/s, v_num=39, train/loss_step=0.040,
    val/loss_step=0.0374, val/loss_epoch=0.0391, train/loss_epoch=0.0396]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.039
Epoch 30: 100%|          | 13/13 [00:02<00:00,  6.41it/s, v_num=39,
    train/loss_step=0.0323, val/loss_step=0.0342, val/loss_epoch=0.0347, train/loss_epoch=0.0352]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.039. Signaling Trainer to stop.
Epoch 30: 100%|          | 13/13 [00:02<00:00,  6.36it/s, v_num=39,
    train/loss_step=0.0323, val/loss_step=0.0342, val/loss_epoch=0.0347, train/loss_epoch=0.0352]

```

Come è possibile analizzare dal grafico proposto in Fig.5.1(a) e dall'output della fase finale di validation, è possibile notare come il modello smetta di convergere fermandosi ad una loss di 0.0347, risultando in immagini ricostruite di scarsissima qualità come anche testimoniato dalle metriche in Tab.5.3.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	11.60 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.2668

Table 5.3: Valori medi di PSNR e SSIM

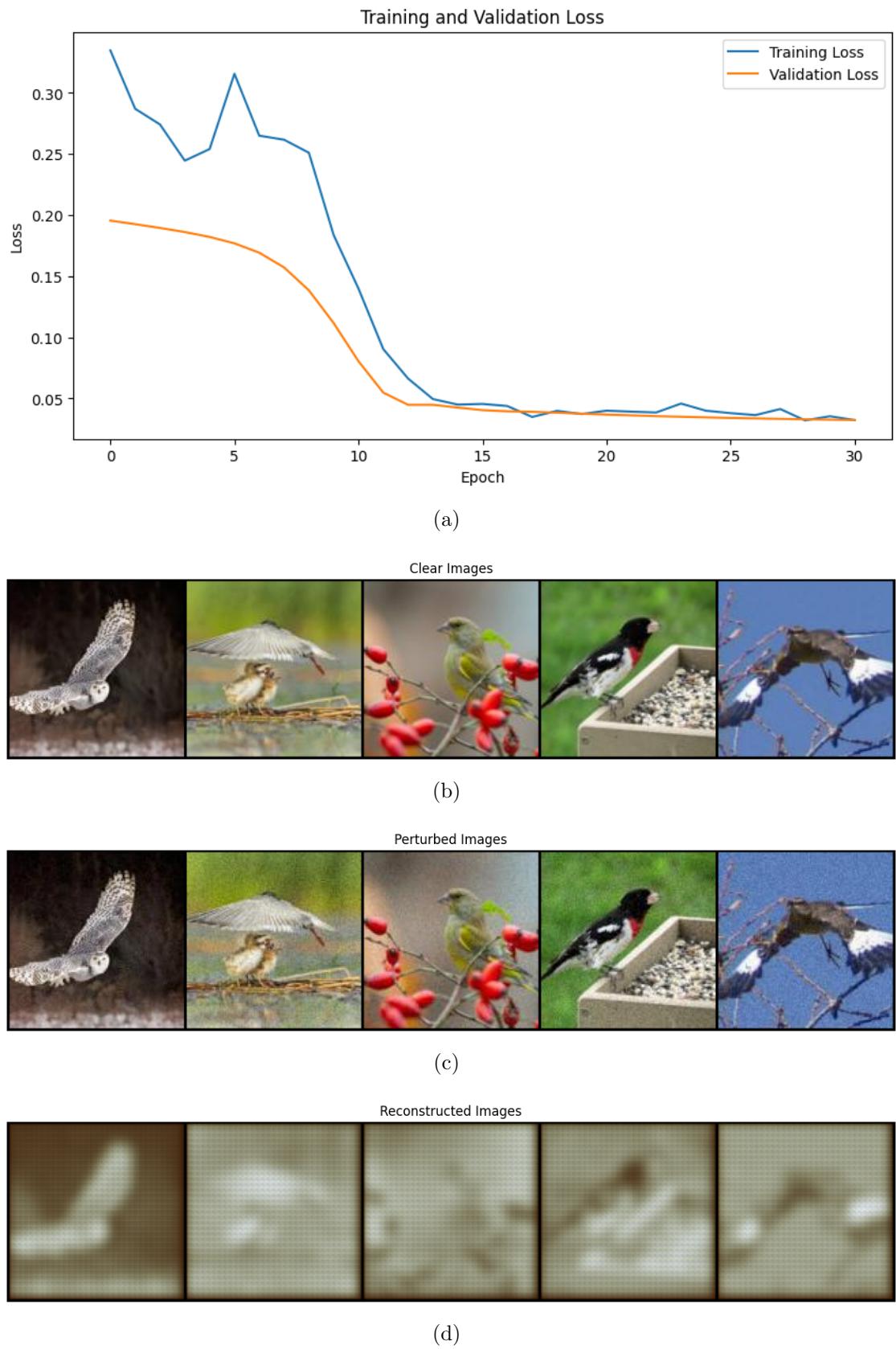


Figure 5.1: Risultati primo esperimento

## 5.2 Secondo esperimento

Layer (type)	Output Shape	Param #
Conv2d-1	[ -1, 8, 128, 128]	224
SiLU-2	[ -1, 8, 128, 128]	0
AvgPool2d-3	[ -1, 8, 64, 64]	0
Conv2d-4	[ -1, 16, 64, 64]	1,168
SiLU-5	[ -1, 16, 64, 64]	0
AvgPool2d-6	[ -1, 16, 32, 32]	0
Conv2d-7	[ -1, 32, 32, 32]	4,640
SiLU-8	[ -1, 32, 32, 32]	0
AvgPool2d-9	[ -1, 32, 16, 16]	0
Conv2d-10	[ -1, 64, 16, 16]	18,496
SiLU-11	[ -1, 64, 16, 16]	0
AvgPool2d-12	[ -1, 64, 8, 8]	0
Conv2d-13	[ -1, 128, 8, 8]	73,856
SiLU-14	[ -1, 128, 8, 8]	0
ConvTranspose2d-15	[ -1, 64, 16, 16]	73,792
SiLU-16	[ -1, 64, 16, 16]	0
ConvTranspose2d-17	[ -1, 32, 32, 32]	18,464
SiLU-18	[ -1, 32, 32, 32]	0
ConvTranspose2d-19	[ -1, 16, 64, 64]	4,624
SiLU-20	[ -1, 16, 64, 64]	0
ConvTranspose2d-21	[ -1, 8, 128, 128]	1,160
SiLU-22	[ -1, 8, 128, 128]	0
Conv2d-23	[ -1, 3, 128, 128]	219
<b>Total params:</b>	196,643	
<b>Trainable params:</b>	196,643	
<b>Non-trainable params:</b>	0	

Table 5.4: Modello secondo esperimento

Qui ho deciso di complicare di più il modello aumentando le fasi di down ed up sampling, oscillando tra 128x128 fino a 8x8, ed i layer convoluzionali risultando in un totale di 196,643 parametri contro i soli 12,035 del primo esperimento.

Output epochs:

```
Epoch 0: 100% | 13/13 [00:02<00:00, 5.92it/s, v_num=40, train/loss_step=0.305,
val/loss_step=0.199, val/loss_epoch=0.301, train/loss_epoch=0.303]
Metric val/loss improved. New best score: 0.301
```

```

Epoch 2: 100%|          13/13 [00:01<00:00,  6.65it/s, v_num=40, train/loss_step=0.287,
      val/loss_step=0.192, val/loss_epoch=0.292, train/loss_epoch=0.293]
Metric val/loss improved by 0.010 >= min_delta = 0.005. New best score: 0.292
Epoch 3: 100%|          13/13 [00:01<00:00,  6.69it/s, v_num=40, train/loss_step=0.283,
      val/loss_step=0.188, val/loss_epoch=0.287, train/loss_epoch=0.288]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.287
Epoch 4: 100%|          13/13 [00:01<00:00,  6.83it/s, v_num=40, train/loss_step=0.269,
      val/loss_step=0.183, val/loss_epoch=0.281, train/loss_epoch=0.283]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.281
Epoch 5: 100%|          13/13 [00:01<00:00,  7.12it/s, v_num=40, train/loss_step=0.257,
      val/loss_step=0.176, val/loss_epoch=0.270, train/loss_epoch=0.275]
Metric val/loss improved by 0.010 >= min_delta = 0.005. New best score: 0.270
Epoch 6: 100%|          13/13 [00:02<00:00,  6.49it/s, v_num=40, train/loss_step=0.243,
      val/loss_step=0.157, val/loss_epoch=0.241, train/loss_epoch=0.259]
Metric val/loss improved by 0.029 >= min_delta = 0.005. New best score: 0.241
Epoch 7: 100%|          13/13 [00:01<00:00,  7.06it/s, v_num=40, train/loss_step=0.225,
      val/loss_step=0.139, val/loss_epoch=0.208, train/loss_epoch=0.224]
Metric val/loss improved by 0.033 >= min_delta = 0.005. New best score: 0.208
Epoch 8: 100%|          13/13 [00:01<00:00,  7.09it/s, v_num=40, train/loss_step=0.161,
      val/loss_step=0.118, val/loss_epoch=0.176, train/loss_epoch=0.192]
Metric val/loss improved by 0.032 >= min_delta = 0.005. New best score: 0.176
Epoch 9: 100%|          13/13 [00:01<00:00,  6.98it/s, v_num=40, train/loss_step=0.176,
      val/loss_step=0.103, val/loss_epoch=0.147, train/loss_epoch=0.162]
Metric val/loss improved by 0.029 >= min_delta = 0.005. New best score: 0.147
Epoch 10: 100%|         13/13 [00:01<00:00,  7.09it/s, v_num=40, train/loss_step=0.118,
      val/loss_step=0.0887, val/loss_epoch=0.122, train/loss_epoch=0.136]
Metric val/loss improved by 0.025 >= min_delta = 0.005. New best score: 0.122
Epoch 11: 100%|         13/13 [00:01<00:00,  7.00it/s, v_num=40, train/loss_step=0.107,
      val/loss_step=0.0762, val/loss_epoch=0.101, train/loss_epoch=0.112]
Metric val/loss improved by 0.021 >= min_delta = 0.005. New best score: 0.101
Epoch 12: 100%|         13/13 [00:01<00:00,  6.75it/s, v_num=40,
      train/loss_step=0.0866, val/loss_step=0.0653, val/loss_epoch=0.0833, train/loss_epoch=0.0926]
Metric val/loss improved by 0.018 >= min_delta = 0.005. New best score: 0.083
Epoch 13: 100%|         13/13 [00:01<00:00,  7.07it/s, v_num=40, train/loss_step=0.068,
      val/loss_step=0.0588, val/loss_epoch=0.0703, train/loss_epoch=0.0771]
Metric val/loss improved by 0.013 >= min_delta = 0.005. New best score: 0.070
Epoch 14: 100%|         13/13 [00:01<00:00,  7.02it/s, v_num=40,
      train/loss_step=0.0534, val/loss_step=0.0519, val/loss_epoch=0.0618, train/loss_epoch=0.0664]
Metric val/loss improved by 0.009 >= min_delta = 0.005. New best score: 0.062
Epoch 16: 100%|         13/13 [00:01<00:00,  6.99it/s, v_num=40,
      train/loss_step=0.0506, val/loss_step=0.0454, val/loss_epoch=0.0542, train/loss_epoch=0.056]
Metric val/loss improved by 0.008 >= min_delta = 0.005. New best score: 0.054
Epoch 20: 100%|         13/13 [00:01<00:00,  7.04it/s, v_num=40,
      train/loss_step=0.0519, val/loss_step=0.0428, val/loss_epoch=0.0483, train/loss_epoch=0.0492]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.048
Epoch 25: 100%|         13/13 [00:01<00:00,  7.10it/s, v_num=40,
      train/loss_step=0.0454, val/loss_step=0.0376, val/loss_epoch=0.0427, train/loss_epoch=0.0437]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.043
Epoch 30: 100%|         13/13 [00:01<00:00,  7.00it/s, v_num=40,
      train/loss_step=0.0466, val/loss_step=0.0359, val/loss_epoch=0.0374, train/loss_epoch=0.0385]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.037
Epoch 40: 100%|         13/13 [00:02<00:00,  5.10it/s, v_num=40,
      train/loss_step=0.0272, val/loss_step=0.0337, val/loss_epoch=0.0334, train/loss_epoch=0.0338]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.037. Signaling Trainer to stop.
Epoch 40: 100%|         13/13 [00:02<00:00,  5.04it/s, v_num=40,
      train/loss_step=0.0272, val/loss_step=0.0337, val/loss_epoch=0.0334, train/loss_epoch=0.0338]

```

Questa scelta ha portato il modello ad avere un leggero peggioramento della loss, nonostante il numero di epoch superiori, e della PSNR.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	12.06 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.2942

Table 5.5: Valori medi di PSNR e SSIM

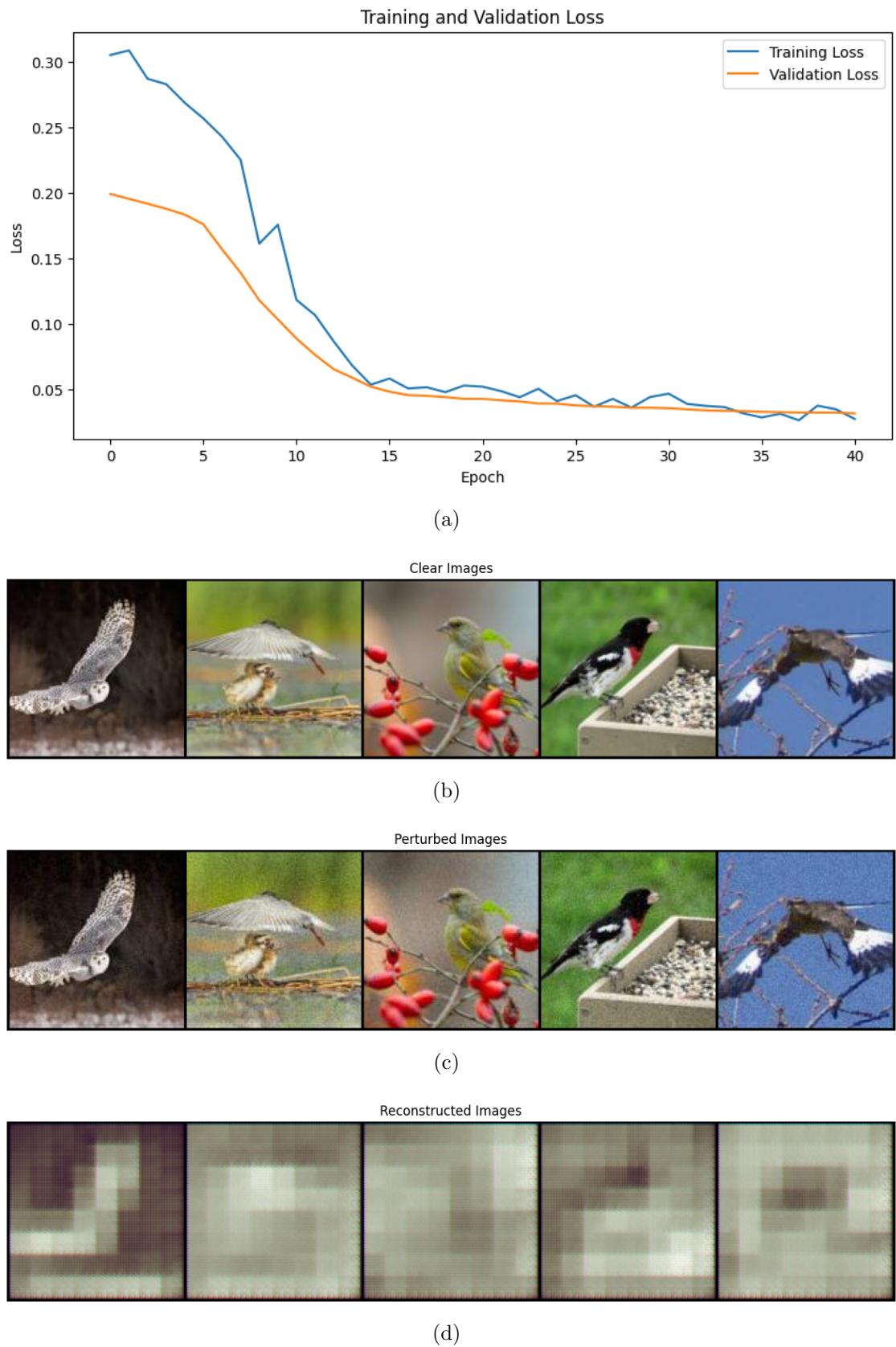


Figure 5.2: Risultati secondo esperimento

### 5.3 Terzo esperimento

Layer (type)	Output Shape	Param #
Conv2d-1	[1, 8, 128, 128]	224
SiLU-2	[1, 8, 128, 128]	0
AvgPool2d-3	[1, 8, 64, 64]	0
Conv2d-4	[1, 16, 64, 64]	1,168
SiLU-5	[1, 16, 64, 64]	0
AvgPool2d-6	[1, 16, 32, 32]	0
Conv2d-7	[1, 32, 32, 32]	4,640
SiLU-8	[1, 32, 32, 32]	0
AvgPool2d-9	[1, 32, 16, 16]	0
Conv2d-10	[1, 64, 16, 16]	18,496
SiLU-11	[1, 64, 16, 16]	0
AvgPool2d-12	[1, 64, 8, 8]	0
Conv2d-13	[1, 128, 8, 8]	73,856
SiLU-14	[1, 128, 8, 8]	0
AvgPool2d-15	[1, 128, 4, 4]	0
Conv2d-16	[1, 256, 4, 4]	295,168
SiLU-17	[1, 256, 4, 4]	0
ConvTranspose2d-18	[1, 128, 8, 8]	295,040
SiLU-19	[1, 128, 8, 8]	0
ConvTranspose2d-20	[1, 64, 16, 16]	73,792
SiLU-21	[1, 64, 16, 16]	0
ConvTranspose2d-22	[1, 32, 32, 32]	18,464
SiLU-23	[1, 32, 32, 32]	0
ConvTranspose2d-24	[1, 16, 64, 64]	4,624
SiLU-25	[1, 16, 64, 64]	0
ConvTranspose2d-26	[1, 8, 128, 128]	1,160
SiLU-27	[1, 8, 128, 128]	0
Conv2d-28	[1, 3, 128, 128]	219
<b>Total params:</b>		786,851
<b>Trainable params:</b>		786,851
<b>Non-trainable params:</b>		0

Table 5.6: Modello terzo esperimento

In questo esperimento ho deciso di aggiungere ulteriori layers di convoluzione e SiLu per verificare se i scarsi risultati ottenuti fin'ora fossero dovuti alla complessità ancora bassa dell'architettura.

Output epochs:

```

Epoch 0: 100%|          13/13 [00:02<00:00,  4.49it/s, v_num=41, train/loss_step=0.341,
      val/loss_step=0.203, val/loss_epoch=0.308, train/loss_epoch=0.310]
Metric val/loss improved. New best score: 0.308
Epoch 1: 100%|          13/13 [00:02<00:00,  6.49it/s, v_num=41, train/loss_step=0.304,
      val/loss_step=0.199, val/loss_epoch=0.303, train/loss_epoch=0.305]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.303
Epoch 2: 100%|          13/13 [00:02<00:00,  6.22it/s, v_num=41, train/loss_step=0.302,
      val/loss_step=0.195, val/loss_epoch=0.298, train/loss_epoch=0.300]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.298
Epoch 3: 100%|          13/13 [00:02<00:00,  5.42it/s, v_num=41, train/loss_step=0.319,
      val/loss_step=0.191, val/loss_epoch=0.292, train/loss_epoch=0.294]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.292
Epoch 4: 100%|          13/13 [00:02<00:00,  4.54it/s, v_num=41, train/loss_step=0.304,
      val/loss_step=0.186, val/loss_epoch=0.286, train/loss_epoch=0.288]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.286
Epoch 5: 100%|          13/13 [00:02<00:00,  6.13it/s, v_num=41, train/loss_step=0.266,
      val/loss_step=0.173, val/loss_epoch=0.269, train/loss_epoch=0.279]
Metric val/loss improved by 0.017 >= min_delta = 0.005. New best score: 0.269
Epoch 6: 100%|          13/13 [00:02<00:00,  6.29it/s, v_num=41, train/loss_step=0.206,
      val/loss_step=0.132, val/loss_epoch=0.203, train/loss_epoch=0.244]
Metric val/loss improved by 0.065 >= min_delta = 0.005. New best score: 0.203
Epoch 7: 100%|          13/13 [00:02<00:00,  6.06it/s, v_num=41, train/loss_step=0.164,
      val/loss_step=0.105, val/loss_epoch=0.155, train/loss_epoch=0.180]
Metric val/loss improved by 0.048 >= min_delta = 0.005. New best score: 0.155
Epoch 8: 100%|          13/13 [00:02<00:00,  6.50it/s, v_num=41, train/loss_step=0.130,
      val/loss_step=0.0913, val/loss_epoch=0.119, train/loss_epoch=0.137]
Metric val/loss improved by 0.036 >= min_delta = 0.005. New best score: 0.119
Epoch 9: 100%|          13/13 [00:02<00:00,  5.58it/s, v_num=41, train/loss_step=0.094,
      val/loss_step=0.0752, val/loss_epoch=0.0956, train/loss_epoch=0.107]
Metric val/loss improved by 0.023 >= min_delta = 0.005. New best score: 0.096
Epoch 10: 100%|         13/13 [00:02<00:00,  6.13it/s, v_num=41, train/loss_step=0.083,
      val/loss_step=0.0665, val/loss_epoch=0.0824, train/loss_epoch=0.0881]
Metric val/loss improved by 0.013 >= min_delta = 0.005. New best score: 0.082
Epoch 11: 100%|         13/13 [00:02<00:00,  6.42it/s, v_num=41,
      train/loss_step=0.0782, val/loss_step=0.0632, val/loss_epoch=0.0747, train/loss_epoch=0.0775]
Metric val/loss improved by 0.008 >= min_delta = 0.005. New best score: 0.075
Epoch 13: 100%|         13/13 [00:01<00:00,  6.63it/s, v_num=41,
      train/loss_step=0.0644, val/loss_step=0.0583, val/loss_epoch=0.0669, train/loss_epoch=0.0676]
Metric val/loss improved by 0.008 >= min_delta = 0.005. New best score: 0.067
Epoch 16: 100%|         13/13 [00:02<00:00,  6.09it/s, v_num=41,
      train/loss_step=0.0454, val/loss_step=0.0537, val/loss_epoch=0.0603, train/loss_epoch=0.0607]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.060
Epoch 19: 100%|         13/13 [00:01<00:00,  6.61it/s, v_num=41,
      train/loss_step=0.0496, val/loss_step=0.0494, val/loss_epoch=0.0553, train/loss_epoch=0.0555]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.055
Epoch 23: 100%|         13/13 [00:02<00:00,  6.41it/s, v_num=41,
      train/loss_step=0.0513, val/loss_step=0.0417, val/loss_epoch=0.0491, train/loss_epoch=0.0497]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.049
Epoch 27: 100%|         13/13 [00:01<00:00,  6.56it/s, v_num=41,
      train/loss_step=0.0504, val/loss_step=0.0373, val/loss_epoch=0.0423, train/loss_epoch=0.0435]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.042
Epoch 37: 100%|         13/13 [00:01<00:00,  6.65it/s, v_num=41,
      train/loss_step=0.0317, val/loss_step=0.0361, val/loss_epoch=0.0374, train/loss_epoch=0.0378]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.042. Signaling Trainer to stop.
Epoch 37: 100%|         13/13 [00:02<00:00,  6.46it/s, v_num=41,
      train/loss_step=0.0317, val/loss_step=0.0361, val/loss_epoch=0.0374, train/loss_epoch=0.0378]

```

Così come nei precedenti due esperimenti non sono riusciti ad ottenere alcun miglioramento, peggiorando ulteriormente le performance.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	11.45 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.2590

Table 5.7: Valori medi di PSNR e SSIM

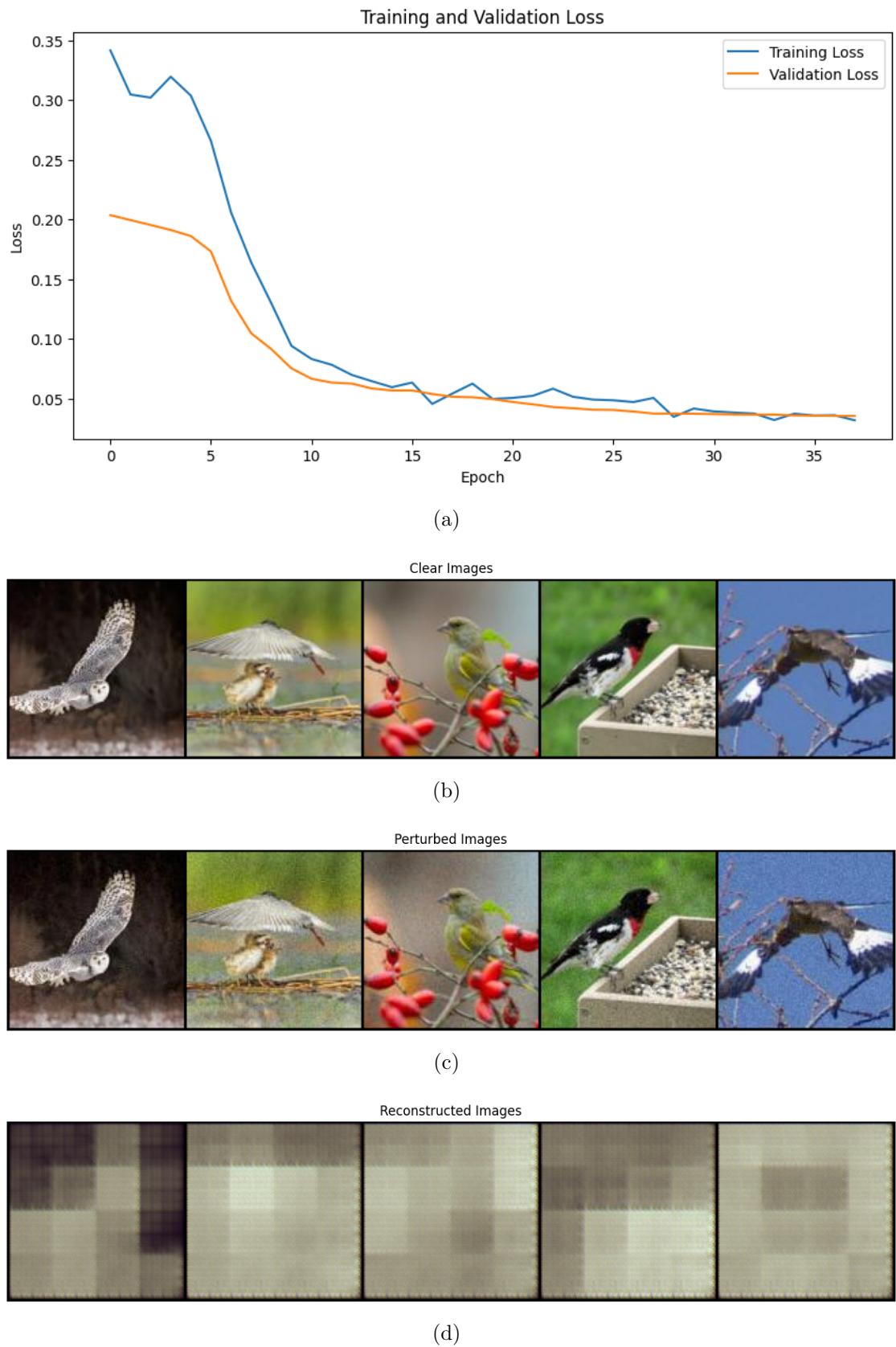


Figure 5.3: Risultati terzo esperimento

## 5.4 Quarto esperimento

Layer (type)	Output Shape	Param #
Conv2d-1	[ -1, 16, 128, 128]	448
BatchNorm2d-2	[ -1, 16, 128, 128]	32
LeakyReLU-3	[ -1, 16, 128, 128]	0
Conv2d-4	[ -1, 32, 128, 128]	4,640
BatchNorm2d-5	[ -1, 32, 128, 128]	64
LeakyReLU-6	[ -1, 32, 128, 128]	0
MaxPool2d-7	[ -1, 32, 64, 64]	0
Conv2d-8	[ -1, 64, 64, 64]	18,496
BatchNorm2d-9	[ -1, 64, 64, 64]	128
LeakyReLU-10	[ -1, 64, 64, 64]	0
Conv2d-11	[ -1, 128, 64, 64]	73,856
BatchNorm2d-12	[ -1, 128, 64, 64]	256
LeakyReLU-13	[ -1, 128, 64, 64]	0
MaxPool2d-14	[ -1, 128, 32, 32]	0
Conv2d-15	[ -1, 256, 32, 32]	295,168
BatchNorm2d-16	[ -1, 256, 32, 32]	512
LeakyReLU-17	[ -1, 256, 32, 32]	0
MaxPool2d-18	[ -1, 256, 16, 16]	0
ConvTranspose2d-19	[ -1, 128, 32, 32]	295,040
BatchNorm2d-20	[ -1, 128, 32, 32]	256
LeakyReLU-21	[ -1, 128, 32, 32]	0
ConvTranspose2d-22	[ -1, 64, 64, 64]	73,792
BatchNorm2d-23	[ -1, 64, 64, 64]	128
LeakyReLU-24	[ -1, 64, 64, 64]	0
ConvTranspose2d-25	[ -1, 32, 128, 128]	18,464
BatchNorm2d-26	[ -1, 32, 128, 128]	64
LeakyReLU-27	[ -1, 32, 128, 128]	0
Conv2d-28	[ -1, 16, 128, 128]	4,624
BatchNorm2d-29	[ -1, 16, 128, 128]	32
LeakyReLU-30	[ -1, 16, 128, 128]	0
Conv2d-31	[ -1, 3, 128, 128]	435
Tanh-32	[ -1, 3, 128, 128]	0
<b>Total params:</b>	786,435	
<b>Trainable params:</b>	786,435	
<b>Non-trainable params:</b>	0	

Table 5.8: Model Architecture

In vista degli scarsi risultati ottenuti negli scorsi esperimenti, nel quarto esperimento ho deciso di stravolgere l'architettura diminuendo l'oscillazione tra dimensioni in [128x128, 16x16] contro i [128x128,8x8] del terzo esperimento e cambiando alcune strategie:

- Ho sostituito gli AvgPool2d con i MaxPool2d
- Ho sostituito le funzioni SiLU con le LeakyReLU con l'obiettivo di migliorare la convergenza
- Ho infine aggiunto dei layer di BatchNormalization per migliorare e stabilizzare il training

Output epochs:

```

Epoch 0: 100%|          | 13/13 [00:04<00:00,  3.19it/s, v_num=42, train/loss_step=0.393,
      val/loss_step=0.241, val/loss_epoch=0.357, train/loss_epoch=0.459]
Metric val/loss improved. New best score: 0.357
Epoch 1: 100%|          | 13/13 [00:03<00:00,  3.31it/s, v_num=42, train/loss_step=0.257,
      val/loss_step=0.207, val/loss_epoch=0.308, train/loss_epoch=0.315]
Metric val/loss improved by 0.049 >= min_delta = 0.005. New best score: 0.308
Epoch 2: 100%|          | 13/13 [00:04<00:00,  3.23it/s, v_num=42, train/loss_step=0.225,
      val/loss_step=0.157, val/loss_epoch=0.231, train/loss_epoch=0.242]
Metric val/loss improved by 0.077 >= min_delta = 0.005. New best score: 0.231
Epoch 3: 100%|          | 13/13 [00:03<00:00,  3.29it/s, v_num=42, train/loss_step=0.178,
      val/loss_step=0.129, val/loss_epoch=0.179, train/loss_epoch=0.198]
Metric val/loss improved by 0.052 >= min_delta = 0.005. New best score: 0.179
Epoch 4: 100%|          | 13/13 [00:04<00:00,  3.19it/s, v_num=42, train/loss_step=0.152,
      val/loss_step=0.108, val/loss_epoch=0.144, train/loss_epoch=0.163]
Metric val/loss improved by 0.035 >= min_delta = 0.005. New best score: 0.144
Epoch 5: 100%|          | 13/13 [00:04<00:00,  3.08it/s, v_num=42, train/loss_step=0.124,
      val/loss_step=0.0822, val/loss_epoch=0.109, train/loss_epoch=0.133]
Metric val/loss improved by 0.035 >= min_delta = 0.005. New best score: 0.109
Epoch 6: 100%|          | 13/13 [00:04<00:00,  3.23it/s, v_num=42, train/loss_step=0.103,
      val/loss_step=0.0667, val/loss_epoch=0.0894, train/loss_epoch=0.108]
Metric val/loss improved by 0.020 >= min_delta = 0.005. New best score: 0.089
Epoch 7: 100%|          | 13/13 [00:04<00:00,  3.15it/s, v_num=42, train/loss_step=0.071,
      val/loss_step=0.0545, val/loss_epoch=0.0719, train/loss_epoch=0.0901]
Metric val/loss improved by 0.017 >= min_delta = 0.005. New best score: 0.072
Epoch 8: 100%|          | 13/13 [00:03<00:00,  3.27it/s, v_num=42, train/loss_step=0.0787,
      val/loss_step=0.0473, val/loss_epoch=0.0627, train/loss_epoch=0.0756]
Metric val/loss improved by 0.009 >= min_delta = 0.005. New best score: 0.063
Epoch 9: 100%|          | 13/13 [00:04<00:00,  3.04it/s, v_num=42, train/loss_step=0.0597,
      val/loss_step=0.0395, val/loss_epoch=0.0532, train/loss_epoch=0.0645]
Metric val/loss improved by 0.010 >= min_delta = 0.005. New best score: 0.053
Epoch 10: 100%|          | 13/13 [00:04<00:00,  3.00it/s, v_num=42,
      train/loss_step=0.0553, val/loss_step=0.035, val/loss_epoch=0.0459, train/loss_epoch=0.0552]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.046
Epoch 12: 100%|          | 13/13 [00:04<00:00,  3.13it/s, v_num=42,
      train/loss_step=0.0369, val/loss_step=0.0277, val/loss_epoch=0.035, train/loss_epoch=0.0415]
Metric val/loss improved by 0.011 >= min_delta = 0.005. New best score: 0.035
Epoch 15: 100%|          | 13/13 [00:04<00:00,  3.20it/s, v_num=42,
      train/loss_step=0.0332, val/loss_step=0.0227, val/loss_epoch=0.026, train/loss_epoch=0.0283]
Metric val/loss improved by 0.009 >= min_delta = 0.005. New best score: 0.026
Epoch 19: 100%|          | 13/13 [00:04<00:00,  3.24it/s, v_num=42,
      train/loss_step=0.0152, val/loss_step=0.0201, val/loss_epoch=0.0204, train/loss_epoch=0.0214]
```

```
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.020
Epoch 29: 100%|                                             | 13/13 [00:04<00:00, 3.25it/s, v_num=42,
train/loss_step=0.0184, val/loss_step=0.0189, val/loss_epoch=0.018, train/loss_epoch=0.0176]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.020. Signaling Trainer to stop.
Epoch 29: 100%|                                             | 13/13 [00:04<00:00, 3.21it/s, v_num=42,
train/loss_step=0.0184, val/loss_step=0.0189, val/loss_epoch=0.018, train/loss_epoch=0.0176]
```

Lo stravolgimento dell'architettura mi ha finalmente permesso di ottenere degli enormi progressi rispetto ai primi esperimenti, riuscendo a raggiungere una MSE loss nell'ultima fase di validation di 0.0189.

Inoltre il miglioramento è ulteriormente visibile nei risultati e metriche in output.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	16.55 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.3638

Table 5.9: Valori medi di PSNR e SSIM

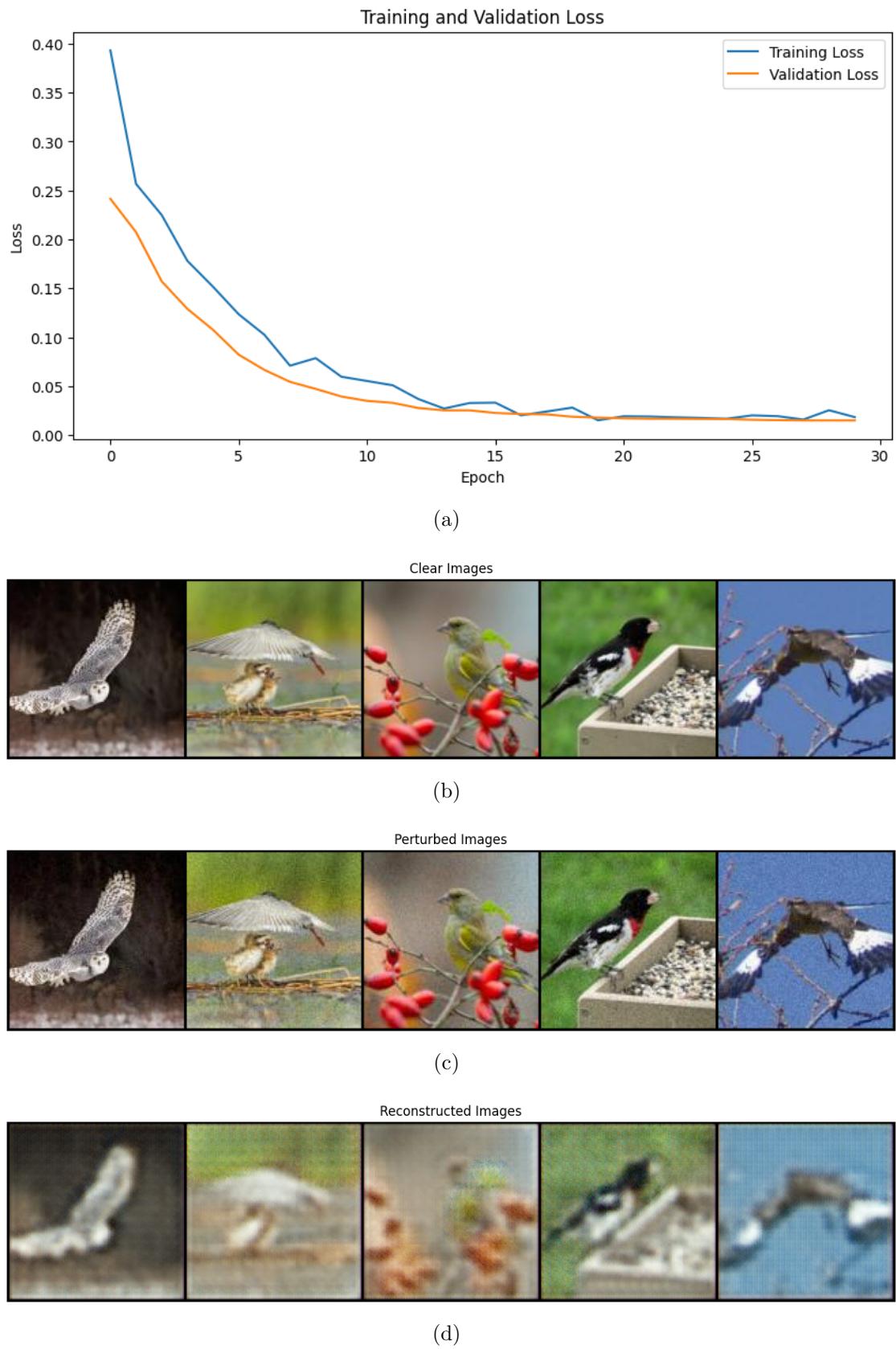


Figure 5.4: Risultati quarto esperimento

## 5.5 Quinto esperimento

Visto il successo della precedente architettura, ho deciso di mantenerne le modifiche fatte aggiungendo ulteriori layers per aumentare la capacità del modello di apprendere dai dati. In tal senso ho aggiunto diversi layer convoluzionali accompagnati da funzioni LeakyReLU e BatchNormalization.

Output epochs:

```

Epoch 0: 100%|          13/13 [00:04<00:00,  2.88it/s, v_num=43, train/loss_step=0.267,
      val/loss_step=0.157, val/loss_epoch=0.244, train/loss_epoch=0.325]
Metric val/loss improved. New best score: 0.244
Epoch 1: 100%|          13/13 [00:04<00:00,  3.00it/s, v_num=43, train/loss_step=0.184,
      val/loss_step=0.124, val/loss_epoch=0.194, train/loss_epoch=0.215]
Metric val/loss improved by 0.050 >= min_delta = 0.005. New best score: 0.194
Epoch 2: 100%|          13/13 [00:04<00:00,  3.00it/s, v_num=43, train/loss_step=0.120,
      val/loss_step=0.101, val/loss_epoch=0.146, train/loss_epoch=0.162]
Metric val/loss improved by 0.048 >= min_delta = 0.005. New best score: 0.146
Epoch 3: 100%|          13/13 [00:04<00:00,  3.01it/s, v_num=43, train/loss_step=0.107,
      val/loss_step=0.0826, val/loss_epoch=0.112, train/loss_epoch=0.125]
Metric val/loss improved by 0.034 >= min_delta = 0.005. New best score: 0.112
Epoch 4: 100%|          13/13 [00:04<00:00,  3.04it/s, v_num=43, train/loss_step=0.106,
      val/loss_step=0.0688, val/loss_epoch=0.0919, train/loss_epoch=0.0984]
Metric val/loss improved by 0.020 >= min_delta = 0.005. New best score: 0.092
Epoch 5: 100%|          13/13 [00:04<00:00,  2.96it/s, v_num=43, train/loss_step=0.0722,
      val/loss_step=0.056, val/loss_epoch=0.072, train/loss_epoch=0.0781]
Metric val/loss improved by 0.020 >= min_delta = 0.005. New best score: 0.072
Epoch 6: 100%|          13/13 [00:04<00:00,  2.93it/s, v_num=43, train/loss_step=0.0636,
      val/loss_step=0.0465, val/loss_epoch=0.0583, train/loss_epoch=0.0639]
Metric val/loss improved by 0.014 >= min_delta = 0.005. New best score: 0.058
Epoch 7: 100%|          13/13 [00:04<00:00,  2.96it/s, v_num=43, train/loss_step=0.0681,
      val/loss_step=0.0378, val/loss_epoch=0.0461, train/loss_epoch=0.0522]
Metric val/loss improved by 0.012 >= min_delta = 0.005. New best score: 0.046
Epoch 8: 100%|          13/13 [00:04<00:00,  2.97it/s, v_num=43, train/loss_step=0.0397,
      val/loss_step=0.0324, val/loss_epoch=0.039, train/loss_epoch=0.0445]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.039
Epoch 10: 100%|         13/13 [00:04<00:00,  2.99it/s, v_num=43,
      train/loss_step=0.0268, val/loss_step=0.0273, val/loss_epoch=0.0322, train/loss_epoch=0.0335]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.032
Epoch 11: 100%|         13/13 [00:04<00:00,  2.95it/s, v_num=43, train/loss_step=0.024,
      val/loss_step=0.0245, val/loss_epoch=0.0271, train/loss_epoch=0.0294]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.027
Epoch 15: 100%|         13/13 [00:04<00:00,  2.96it/s, v_num=43,
      train/loss_step=0.0194, val/loss_step=0.0206, val/loss_epoch=0.0217, train/loss_epoch=0.0222]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.022
Epoch 25: 100%|         13/13 [00:04<00:00,  2.76it/s, v_num=43,
      train/loss_step=0.0159, val/loss_step=0.0186, val/loss_epoch=0.0177, train/loss_epoch=0.0176]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.022. Signaling Trainer to stop.
Epoch 25: 100%|         13/13 [00:04<00:00,  2.73it/s, v_num=43,
      train/loss_step=0.0159, val/loss_step=0.0186, val/loss_epoch=0.0177, train/loss_epoch=0.0176]
```

Le modifiche hanno portato ad un ulteriore miglioramento come visibile dalle immagini ricostruire Fig.5.5(d) e dalle metriche PSNR e SSIM Tab.5.11

Layer (type)	Output Shape	Param #
Conv2d-1	[ -1, 16, 128, 128]	448
BatchNorm2d-2	[ -1, 16, 128, 128]	32
LeakyReLU-3	[ -1, 16, 128, 128]	0
Conv2d-4	[ -1, 32, 128, 128]	4,640
BatchNorm2d-5	[ -1, 32, 128, 128]	64
LeakyReLU-6	[ -1, 32, 128, 128]	0
MaxPool2d-7	[ -1, 32, 64, 64]	0
Conv2d-8	[ -1, 64, 64, 64]	18,496
BatchNorm2d-9	[ -1, 64, 64, 64]	128
LeakyReLU-10	[ -1, 64, 64, 64]	0
Conv2d-11	[ -1, 128, 64, 64]	73,856
BatchNorm2d-12	[ -1, 128, 64, 64]	256
LeakyReLU-13	[ -1, 128, 64, 64]	0
MaxPool2d-14	[ -1, 128, 32, 32]	0
Conv2d-15	[ -1, 128, 32, 32]	147,584
BatchNorm2d-16	[ -1, 128, 32, 32]	256
LeakyReLU-17	[ -1, 128, 32, 32]	0
Conv2d-18	[ -1, 256, 32, 32]	295,168
BatchNorm2d-19	[ -1, 256, 32, 32]	512
LeakyReLU-20	[ -1, 256, 32, 32]	0
MaxPool2d-21	[ -1, 256, 16, 16]	0
ConvTranspose2d-22	[ -1, 128, 32, 32]	295,040
BatchNorm2d-23	[ -1, 128, 32, 32]	256
LeakyReLU-24	[ -1, 128, 32, 32]	0
Conv2d-25	[ -1, 128, 32, 32]	147,584
BatchNorm2d-26	[ -1, 128, 32, 32]	256
LeakyReLU-27	[ -1, 128, 32, 32]	0
ConvTranspose2d-28	[ -1, 64, 64, 64]	73,792
BatchNorm2d-29	[ -1, 64, 64, 64]	128
LeakyReLU-30	[ -1, 64, 64, 64]	0
ConvTranspose2d-31	[ -1, 32, 128, 128]	18,464
BatchNorm2d-32	[ -1, 32, 128, 128]	64
LeakyReLU-33	[ -1, 32, 128, 128]	0
Conv2d-34	[ -1, 16, 128, 128]	4,624
BatchNorm2d-35	[ -1, 16, 128, 128]	32
LeakyReLU-36	[ -1, 16, 128, 128]	0
Conv2d-37	[ -1, 3, 128, 128]	435
Tanh-38	[ -1, 3, 128, 128]	0
<b>Total params:</b>	1,082,115	
<b>Trainable params:</b>	1,082,115	
<b>Non-trainable params:</b>	0	

Table 5.10: Modello quinto esperimento

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	16.41 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.3443

Table 5.11: Valori medi di PSNR e SSIM

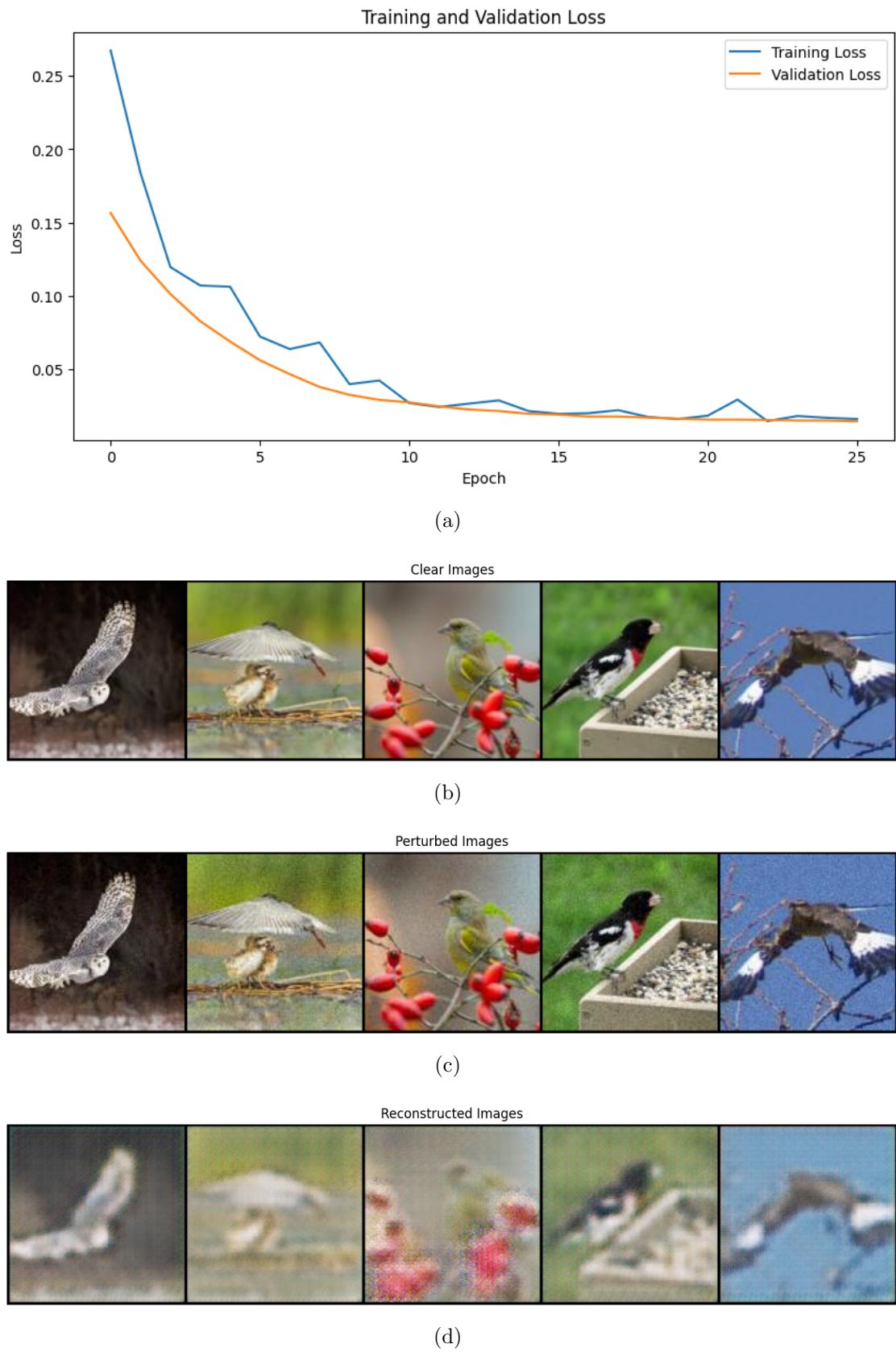


Figure 5.5: Risultati quinto esperimento

## 5.6 Sesto esperimento

Table 5.12: Modello sesto esperimento

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
Conv2d-1	[-1, 16, 128, 128]	448
BatchNorm2d-2	[-1, 16, 128, 128]	32
LeakyReLU-3	[-1, 16, 128, 128]	0
Conv2d-4	[-1, 32, 128, 128]	4,640
BatchNorm2d-5	[-1, 32, 128, 128]	64
LeakyReLU-6	[-1, 32, 128, 128]	0
MaxPool2d-7	[-1, 32, 64, 64]	0
Conv2d-8	[-1, 64, 64, 64]	18,496
BatchNorm2d-9	[-1, 64, 64, 64]	128
LeakyReLU-10	[-1, 64, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	73,856
BatchNorm2d-12	[-1, 128, 64, 64]	256
LeakyReLU-13	[-1, 128, 64, 64]	0
MaxPool2d-14	[-1, 128, 32, 32]	0
Conv2d-15	[-1, 128, 32, 32]	147,584
BatchNorm2d-16	[-1, 128, 32, 32]	256
LeakyReLU-17	[-1, 128, 32, 32]	0
Conv2d-18	[-1, 128, 32, 32]	147,584
BatchNorm2d-19	[-1, 128, 32, 32]	256
LeakyReLU-20	[-1, 128, 32, 32]	0
Conv2d-21	[-1, 256, 32, 32]	295,168
BatchNorm2d-22	[-1, 256, 32, 32]	512
LeakyReLU-23	[-1, 256, 32, 32]	0
MaxPool2d-24	[-1, 256, 16, 16]	0
ConvTranspose2d-25	[-1, 128, 32, 32]	295,040
BatchNorm2d-26	[-1, 128, 32, 32]	256
LeakyReLU-27	[-1, 128, 32, 32]	0
Conv2d-28	[-1, 128, 32, 32]	147,584

Table 5.12: Modello sesto esperimento (continua)

Layer (type)	Output Shape	Param #
BatchNorm2d-29	[-1, 128, 32, 32]	256
LeakyReLU-30	[-1, 128, 32, 32]	0
Conv2d-31	[-1, 128, 32, 32]	147,584
BatchNorm2d-32	[-1, 128, 32, 32]	256
LeakyReLU-33	[-1, 128, 32, 32]	0
ConvTranspose2d-34	[-1, 64, 64, 64]	73,792
BatchNorm2d-35	[-1, 64, 64, 64]	128
LeakyReLU-36	[-1, 64, 64, 64]	0
ConvTranspose2d-37	[-1, 32, 128, 128]	18,464
BatchNorm2d-38	[-1, 32, 128, 128]	64
LeakyReLU-39	[-1, 32, 128, 128]	0
Conv2d-40	[-1, 16, 128, 128]	4,624
BatchNorm2d-41	[-1, 16, 128, 128]	32
LeakyReLU-42	[-1, 16, 128, 128]	0
Conv2d-43	[-1, 3, 128, 128]	435
Tanh-44	[-1, 3, 128, 128]	0
Total params:	1,377,795	
Trainable params:	1,377,795	
Non-trainable params:	0	

Visto il miglioramento della scorsa architettura, ho deciso di continuare con la stessa filosia aumentandone la complessità con la stessa strategia.

Output epochs:

```

Epoch 0: 100%| 13/13 [00:06<00:00,  2.07it/s, v_num=44, train/loss_step=0.191,
      val/loss_step=0.168, val/loss_epoch=0.260, train/loss_epoch=0.258]
Metric val/loss improved. New best score: 0.260
Epoch 1: 100%| 13/13 [00:06<00:00,  2.11it/s, v_num=44, train/loss_step=0.125,
      val/loss_step=0.121, val/loss_epoch=0.192, train/loss_epoch=0.152]
Metric val/loss improved by 0.068 >= min_delta = 0.005. New best score: 0.192
Epoch 2: 100%| 13/13 [00:06<00:00,  2.12it/s, v_num=44, train/loss_step=0.0936,
      val/loss_step=0.0682, val/loss_epoch=0.106, train/loss_epoch=0.104]
Metric val/loss improved by 0.086 >= min_delta = 0.005. New best score: 0.106
Epoch 3: 100%| 13/13 [00:06<00:00,  2.12it/s, v_num=44, train/loss_step=0.064,
      val/loss_step=0.050, val/loss_epoch=0.0698, train/loss_epoch=0.0745]
Metric val/loss improved by 0.036 >= min_delta = 0.005. New best score: 0.070

```

```

Epoch 4: 100%|          13/13 [00:06<00:00,  2.09it/s, v_num=44, train/loss_step=0.0461,
      val/loss_step=0.0366, val/loss_epoch=0.0503, train/loss_epoch=0.0561]
Metric val/loss improved by 0.020 >= min_delta = 0.005. New best score: 0.050
Epoch 5: 100%|          13/13 [00:06<00:00,  2.09it/s, v_num=44, train/loss_step=0.0355,
      val/loss_step=0.0291, val/loss_epoch=0.039, train/loss_epoch=0.0441]
Metric val/loss improved by 0.011 >= min_delta = 0.005. New best score: 0.039
Epoch 6: 100%|          13/13 [00:06<00:00,  2.07it/s, v_num=44, train/loss_step=0.0298,
      val/loss_step=0.0262, val/loss_epoch=0.0333, train/loss_epoch=0.0366]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.033
Epoch 8: 100%|          13/13 [00:06<00:00,  2.15it/s, v_num=44, train/loss_step=0.0296,
      val/loss_step=0.0228, val/loss_epoch=0.0264, train/loss_epoch=0.0278]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.026
Epoch 11: 100%|         13/13 [00:06<00:00,  2.11it/s, v_num=44,
      train/loss_step=0.0249, val/loss_step=0.0202, val/loss_epoch=0.0208, train/loss_epoch=0.022]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.021
Epoch 21: 100%|         13/13 [00:06<00:00,  2.10it/s, v_num=44,
      train/loss_step=0.0184, val/loss_step=0.0203, val/loss_epoch=0.0193, train/loss_epoch=0.0181]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.021. Signaling Trainer to stop.
Epoch 21: 100%|         13/13 [00:06<00:00,  2.07it/s, v_num=44,
      train/loss_step=0.0184, val/loss_step=0.0203, val/loss_epoch=0.0193, train/loss_epoch=0.0181]

```

Purtroppo, a differenza delle aspettative, l'aumento di complessità non ha portato ad un miglioramento, ma bensì ad un leggero peggioramento.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	15.11 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.3251

Table 5.13: Valori medi di PSNR e SSIM

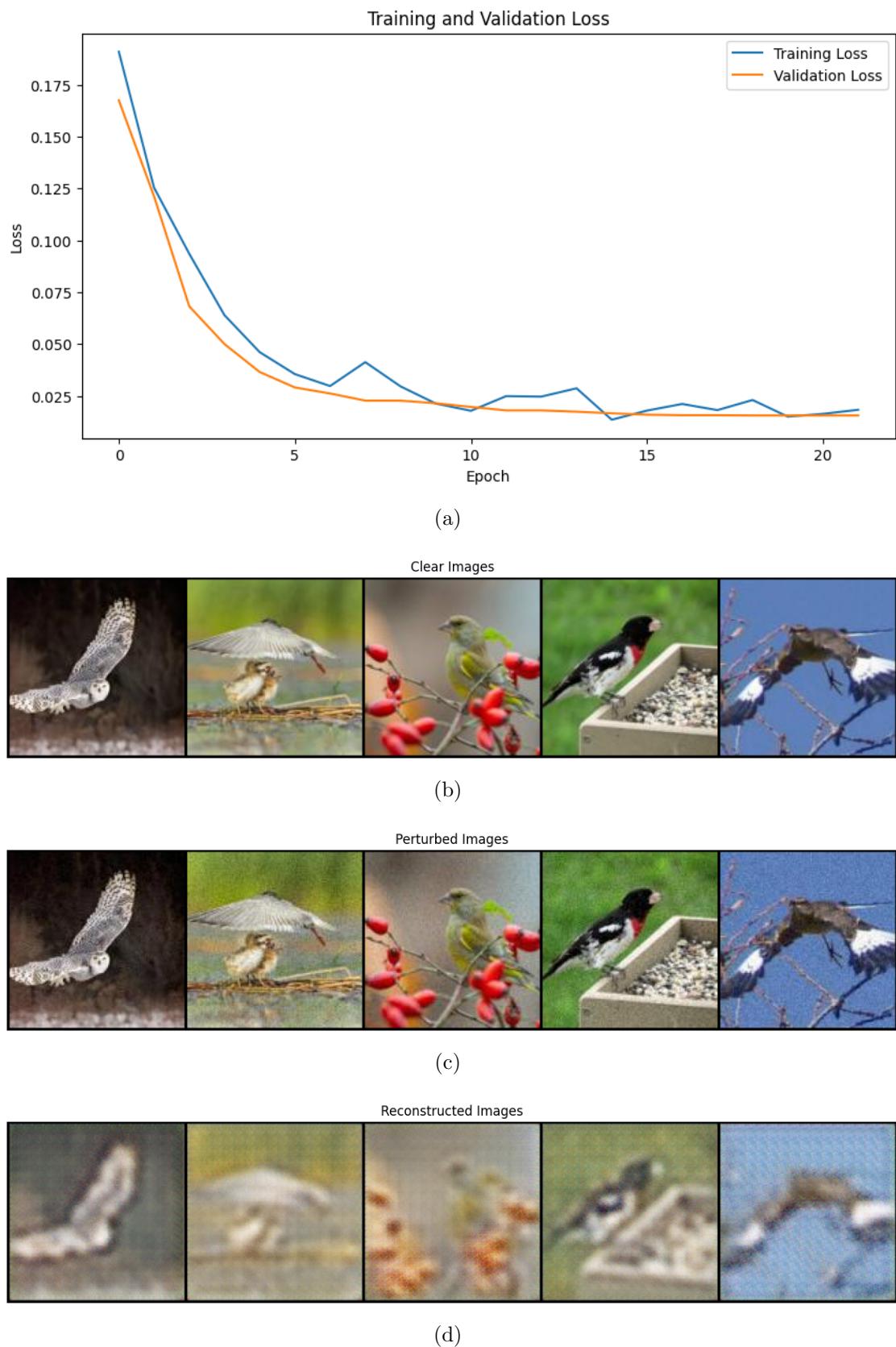


Figure 5.6: Risultati sesto esperimento

## 5.7 Settimo esperimento

Table 5.14: Modello settimo esperimento

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
Conv2d-1	[-1, 16, 128, 128]	448
BatchNorm2d-2	[-1, 16, 128, 128]	32
LeakyReLU-3	[-1, 16, 128, 128]	0
Conv2d-4	[-1, 32, 128, 128]	4,640
BatchNorm2d-5	[-1, 32, 128, 128]	64
LeakyReLU-6	[-1, 32, 128, 128]	0
MaxPool2d-7	[-1, 32, 64, 64]	0
Conv2d-8	[-1, 64, 64, 64]	18,496
BatchNorm2d-9	[-1, 64, 64, 64]	128
LeakyReLU-10	[-1, 64, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	73,856
BatchNorm2d-12	[-1, 128, 64, 64]	256
LeakyReLU-13	[-1, 128, 64, 64]	0
MaxPool2d-14	[-1, 128, 32, 32]	0
Conv2d-15	[-1, 128, 32, 32]	147,584
BatchNorm2d-16	[-1, 128, 32, 32]	256
LeakyReLU-17	[-1, 128, 32, 32]	0
Conv2d-18	[-1, 256, 32, 32]	295,168
BatchNorm2d-19	[-1, 256, 32, 32]	512
LeakyReLU-20	[-1, 256, 32, 32]	0
Conv2d-21	[-1, 256, 32, 32]	590,080
BatchNorm2d-22	[-1, 256, 32, 32]	512
LeakyReLU-23	[-1, 256, 32, 32]	0
ConvTranspose2d-24	[-1, 256, 64, 64]	590,080
BatchNorm2d-25	[-1, 256, 64, 64]	512
LeakyReLU-26	[-1, 256, 64, 64]	0
ConvTranspose2d-27	[-1, 128, 128, 128]	295,040
BatchNorm2d-28	[-1, 128, 128, 128]	256

Table 5.14: Modello settimo esperimento (continua)

Layer (type)	Output Shape	Param #
LeakyReLU-29	[-1, 128, 128, 128]	0
Conv2d-30	[-1, 128, 128, 128]	147,584
BatchNorm2d-31	[-1, 128, 128, 128]	256
LeakyReLU-32	[-1, 128, 128, 128]	0
Conv2d-33	[-1, 64, 128, 128]	73,792
BatchNorm2d-34	[-1, 64, 128, 128]	128
LeakyReLU-35	[-1, 64, 128, 128]	0
Conv2d-36	[-1, 32, 128, 128]	18,464
BatchNorm2d-37	[-1, 32, 128, 128]	64
LeakyReLU-38	[-1, 32, 128, 128]	0
Conv2d-39	[-1, 16, 128, 128]	4,624
BatchNorm2d-40	[-1, 16, 128, 128]	32
LeakyReLU-41	[-1, 16, 128, 128]	0
Conv2d-42	[-1, 3, 128, 128]	435
Tanh-43	[-1, 3, 128, 128]	0
Total params:	2,263,299	
Trainable params:	2,263,299	
Non-trainable params:	0	

Nonostante l'insuccesso della scorsa architettura, ho deciso lo stesso di aumentare ulteriormente la complessità del modello raddoppiandone quasi i parametri.

Tuttavia ho anche deciso di limitare il down e up sampling di 1 fattore con l'idea di non perdere troppe informazioni durante la fase di encoding al fine di garantire al modello di poter apprendere più informazioni.

Output epochs:

```

Epoch 0: 100%| 13/13 [04:29<00:00,  0.05it/s, v_num=45, train/loss_step=0.149,
val/loss_step=0.169, val/loss_epoch=0.264, train/loss_epoch=0.231]
Metric val/loss improved. New best score: 0.264
Epoch 1: 100%| 13/13 [04:12<00:00,  0.05it/s, v_num=45, train/loss_step=0.109,
val/loss_step=0.115, val/loss_epoch=0.183, train/loss_epoch=0.124]

```

```

Metric val/loss improved by 0.081 >= min_delta = 0.005. New best score: 0.183
Epoch 2: 100%|                                         | 13/13 [03:52<00:00, 0.06it/s, v_num=45, train/loss_step=0.0884,
    val/loss_step=0.0603, val/loss_epoch=0.0984, train/loss_epoch=0.0898]
Metric val/loss improved by 0.085 >= min_delta = 0.005. New best score: 0.098
Epoch 3: 100%|                                         | 13/13 [03:56<00:00, 0.05it/s, v_num=45, train/loss_step=0.0702,
    val/loss_step=0.0406, val/loss_epoch=0.0618, train/loss_epoch=0.0676]
Metric val/loss improved by 0.037 >= min_delta = 0.005. New best score: 0.062
Epoch 4: 100%|                                         | 13/13 [03:58<00:00, 0.05it/s, v_num=45, train/loss_step=0.0559,
    val/loss_step=0.0294, val/loss_epoch=0.0433, train/loss_epoch=0.0528]
Metric val/loss improved by 0.018 >= min_delta = 0.005. New best score: 0.043
Epoch 5: 100%|                                         | 13/13 [03:54<00:00, 0.06it/s, v_num=45, train/loss_step=0.035,
    val/loss_step=0.0284, val/loss_epoch=0.0357, train/loss_epoch=0.0424]
Metric val/loss improved by 0.008 >= min_delta = 0.005. New best score: 0.036
Epoch 6: 100%|                                         | 13/13 [03:56<00:00, 0.06it/s, v_num=45, train/loss_step=0.0249,
    val/loss_step=0.0254, val/loss_epoch=0.0298, train/loss_epoch=0.0354]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.030
Epoch 8: 100%|                                         | 13/13 [03:54<00:00, 0.06it/s, v_num=45, train/loss_step=0.0204,
    val/loss_step=0.0191, val/loss_epoch=0.0225, train/loss_epoch=0.0246]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.023
Epoch 13: 100%|                                         | 13/13 [03:55<00:00, 0.06it/s, v_num=45,
    train/loss_step=0.0162, val/loss_step=0.0173, val/loss_epoch=0.0166, train/loss_epoch=0.0169]
Metric val/loss improved by 0.006 >= min_delta = 0.005. New best score: 0.017
Epoch 23: 100%|                                         | 13/13 [04:07<00:00, 0.05it/s, v_num=45,
    train/loss_step=0.0126, val/loss_step=0.0164, val/loss_epoch=0.0146, train/loss_epoch=0.0144]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.017. Signaling Trainer to stop.
Epoch 23: 100%|                                         | 13/13 [04:07<00:00, 0.05it/s, v_num=45,
    train/loss_step=0.0126, val/loss_step=0.0164, val/loss_epoch=0.0146, train/loss_epoch=0.0144]

```

Queste modifiche hanno leggermente migliorato la MSE loss, ma il miglioramento principale è in particolar modo visibile nella metrica SSIM nonché dal risultato vivido.

Tuttavia i risultati restano ancora molto lontani da quelli sperati.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	17.72 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.4765

Table 5.15: Valori medi di PSNR e SSIM

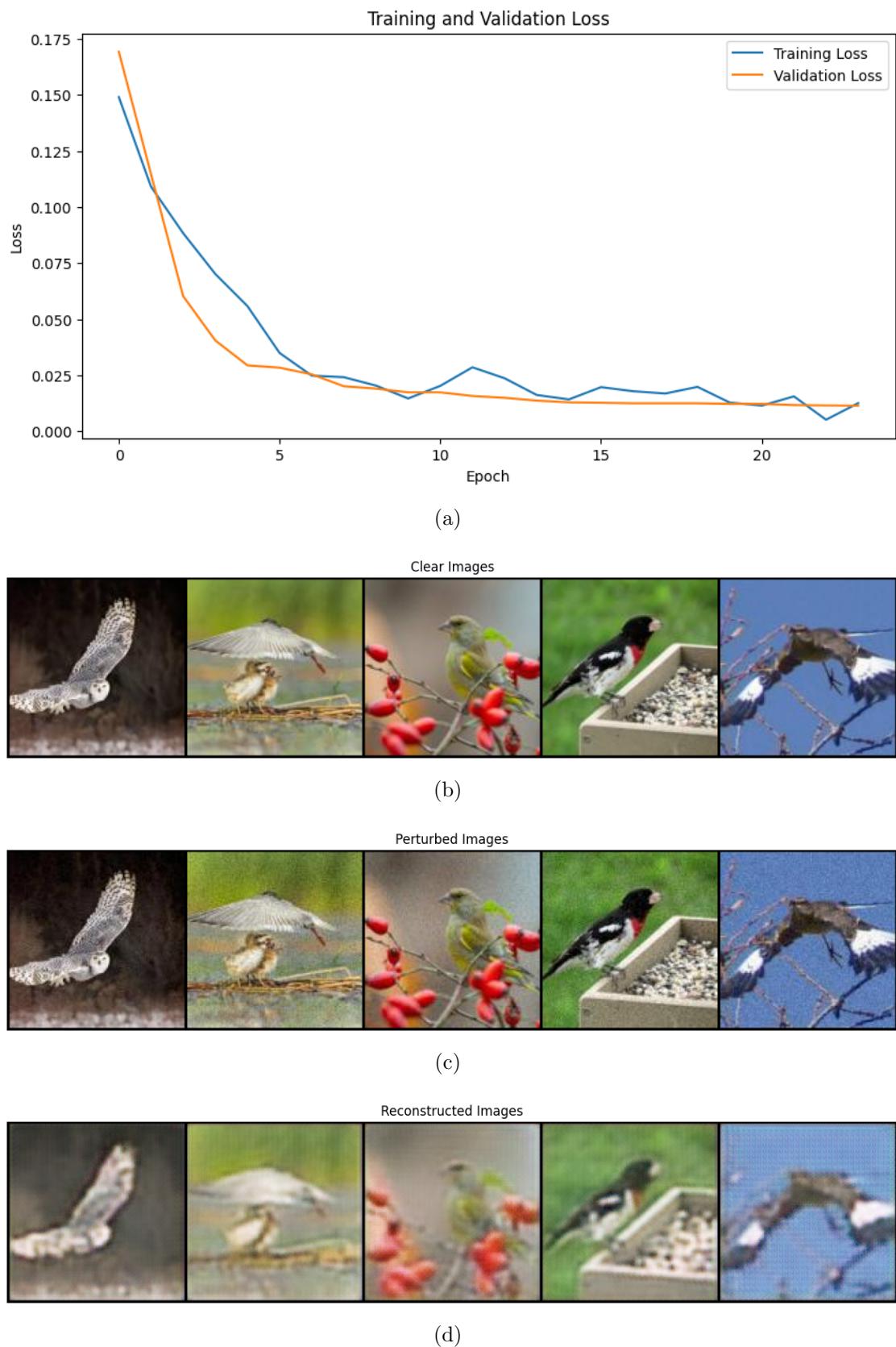


Figure 5.7: Risultati settimo esperimento

## 5.8 Ottavo esperimento

Table 5.16: Modello ottavo esperimento

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
Conv2d-1	[-1, 16, 128, 128]	448
BatchNorm2d-2	[-1, 16, 128, 128]	32
LeakyReLU-3	[-1, 16, 128, 128]	0
Conv2d-4	[-1, 32, 128, 128]	4,640
BatchNorm2d-5	[-1, 32, 128, 128]	64
LeakyReLU-6	[-1, 32, 128, 128]	0
MaxPool2d-7	[-1, 32, 64, 64]	0
Conv2d-8	[-1, 64, 64, 64]	18,496
BatchNorm2d-9	[-1, 64, 64, 64]	128
LeakyReLU-10	[-1, 64, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	73,856
BatchNorm2d-12	[-1, 128, 64, 64]	256
LeakyReLU-13	[-1, 128, 64, 64]	0
Conv2d-14	[-1, 128, 64, 64]	147,584
BatchNorm2d-15	[-1, 128, 64, 64]	256
LeakyReLU-16	[-1, 128, 64, 64]	0
Conv2d-17	[-1, 256, 64, 64]	295,168
BatchNorm2d-18	[-1, 256, 64, 64]	512
LeakyReLU-19	[-1, 256, 64, 64]	0
MaxPool2d-20	[-1, 256, 32, 32]	0
Conv2d-21	[-1, 256, 32, 32]	590,080
BatchNorm2d-22	[-1, 256, 32, 32]	512
LeakyReLU-23	[-1, 256, 32, 32]	0
ConvTranspose2d-24	[-1, 256, 64, 64]	590,080
BatchNorm2d-25	[-1, 256, 64, 64]	512
LeakyReLU-26	[-1, 256, 64, 64]	0
ConvTranspose2d-27	[-1, 128, 128, 128]	295,040
BatchNorm2d-28	[-1, 128, 128, 128]	256

Table 5.16: Modello ottavo esperimento (continua)

Layer (type)	Output Shape	Param #
LeakyReLU-29	[-1, 128, 128, 128]	0
Conv2d-30	[-1, 128, 128, 128]	147,584
BatchNorm2d-31	[-1, 128, 128, 128]	256
LeakyReLU-32	[-1, 128, 128, 128]	0
Conv2d-33	[-1, 64, 128, 128]	73,792
BatchNorm2d-34	[-1, 64, 128, 128]	128
LeakyReLU-35	[-1, 64, 128, 128]	0
Conv2d-36	[-1, 32, 128, 128]	18,464
BatchNorm2d-37	[-1, 32, 128, 128]	64
LeakyReLU-38	[-1, 32, 128, 128]	0
Conv2d-39	[-1, 16, 128, 128]	4,624
BatchNorm2d-40	[-1, 16, 128, 128]	32
LeakyReLU-41	[-1, 16, 128, 128]	0
Conv2d-42	[-1, 3, 128, 128]	435
Tanh-43	[-1, 3, 128, 128]	0
Total params:	2,263,299	
Trainable params:	2,263,299	
Non-trainable params:	0	

Dato il leggero successo della scorsa architettura, con questo esperimento ho deciso di non aumentare ulteriormente la complessità ma cambiare la disposizione dei layer di convoluzione concentrandoli di più nella fase di "alta risoluzione" del dato prima che avvenga il down-sampling. Questo è motivato dall'obiettivo di far cogliere maggiori dettagli al modello in modo tale da permettere una migliore rappresentazione che verrà poi usata dalla fase di decoding.

Output epochs:

```

Epoch 0: 100%| 13/13 [04:16<00:00,  0.05it/s, v_num=46, train/loss_step=0.151,
val/loss_step=0.154, val/loss_epoch=0.245, train/loss_epoch=0.233]
Metric val/loss improved. New best score: 0.245
Epoch 1: 100%| 13/13 [04:30<00:00,  0.05it/s, v_num=46, train/loss_step=0.114,
val/loss_step=0.118, val/loss_epoch=0.180, train/loss_epoch=0.140]

```

```

Metric val/loss improved by 0.064 >= min_delta = 0.005. New best score: 0.180
Epoch 2: 100%|                                         | 13/13 [04:15<00:00, 0.05it/s, v_num=46, train/loss_step=0.0919,
    val/loss_step=0.0791, val/loss_epoch=0.117, train/loss_epoch=0.102]
Metric val/loss improved by 0.063 >= min_delta = 0.005. New best score: 0.117
Epoch 3: 100%|                                         | 13/13 [04:12<00:00, 0.05it/s, v_num=46, train/loss_step=0.073,
    val/loss_step=0.0515, val/loss_epoch=0.0787, train/loss_epoch=0.0751]
Metric val/loss improved by 0.039 >= min_delta = 0.005. New best score: 0.079
Epoch 4: 100%|                                         | 13/13 [04:12<00:00, 0.05it/s, v_num=46, train/loss_step=0.0498,
    val/loss_step=0.0374, val/loss_epoch=0.0526, train/loss_epoch=0.0557]
Metric val/loss improved by 0.026 >= min_delta = 0.005. New best score: 0.053
Epoch 5: 100%|                                         | 13/13 [04:12<00:00, 0.05it/s, v_num=46, train/loss_step=0.0299,
    val/loss_step=0.027, val/loss_epoch=0.0372, train/loss_epoch=0.0418]
Metric val/loss improved by 0.015 >= min_delta = 0.005. New best score: 0.037
Epoch 6: 100%|                                         | 13/13 [04:12<00:00, 0.05it/s, v_num=46, train/loss_step=0.0386,
    val/loss_step=0.0255, val/loss_epoch=0.0319, train/loss_epoch=0.0325]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.032
Epoch 7: 100%|                                         | 13/13 [04:12<00:00, 0.05it/s, v_num=46, train/loss_step=0.0232,
    val/loss_step=0.0204, val/loss_epoch=0.024, train/loss_epoch=0.0266]
Metric val/loss improved by 0.008 >= min_delta = 0.005. New best score: 0.024
Epoch 10: 100%|                                         | 13/13 [04:12<00:00, 0.05it/s, v_num=46,
    train/loss_step=0.0135, val/loss_step=0.0185, val/loss_epoch=0.0189, train/loss_epoch=0.0186]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.019
Epoch 20: 100%|                                         | 13/13 [04:13<00:00, 0.05it/s, v_num=46,
    train/loss_step=0.0187, val/loss_step=0.0176, val/loss_epoch=0.015, train/loss_epoch=0.0146]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.019. Signaling Trainer to stop.
Epoch 20: 100%|                                         | 13/13 [04:13<00:00, 0.05it/s, v_num=46,
    train/loss_step=0.0187, val/loss_step=0.0176, val/loss_epoch=0.015, train/loss_epoch=0.0146]

```

Purtroppo le modifiche non hanno portato ad alcun miglioramento, mantenendo gli stessi risultati dell'architettura precedente.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	17.83 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.4713

Table 5.17: Valori medi di PSNR e SSIM

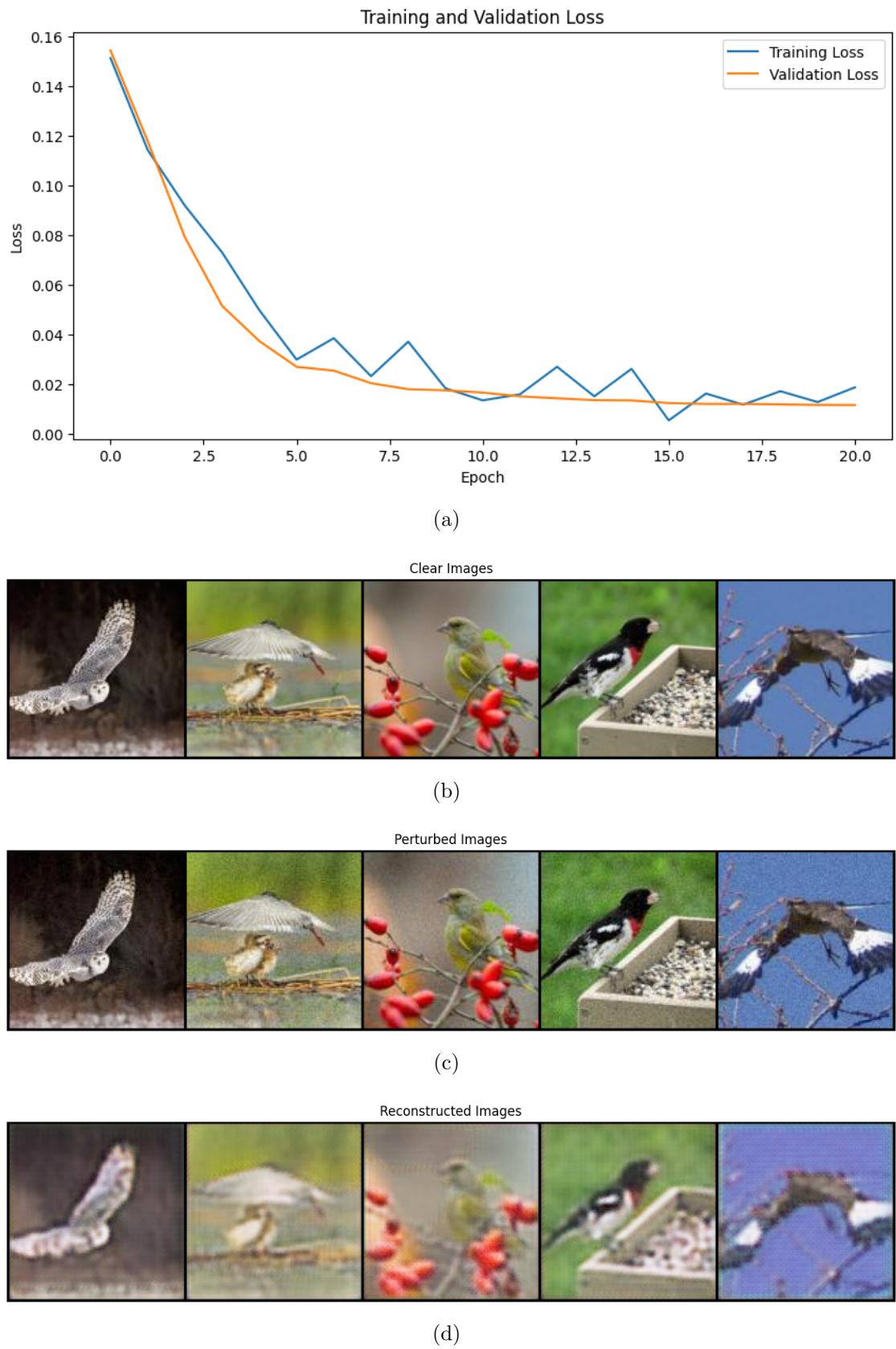


Figure 5.8: Risultati ottavo esperimento

## 5.9 Nono esperimento

Table 5.18: Modello nono esperimento

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
Conv2d-1	[ -1, 16, 130, 130]	2,368
BatchNorm2d-2	[ -1, 16, 130, 130]	32
LeakyReLU-3	[ -1, 16, 130, 130]	0
Conv2d-4	[ -1, 32, 130, 130]	12,832
BatchNorm2d-5	[ -1, 32, 130, 130]	64
LeakyReLU-6	[ -1, 32, 130, 130]	0
MaxPool2d-7	[ -1, 32, 65, 65]	0
Conv2d-8	[ -1, 64, 65, 65]	18,496
BatchNorm2d-9	[ -1, 64, 65, 65]	128
LeakyReLU-10	[ -1, 64, 65, 65]	0
Conv2d-11	[ -1, 128, 65, 65]	73,856
BatchNorm2d-12	[ -1, 128, 65, 65]	256
LeakyReLU-13	[ -1, 128, 65, 65]	0
Conv2d-14	[ -1, 128, 65, 65]	147,584
BatchNorm2d-15	[ -1, 128, 65, 65]	256
LeakyReLU-16	[ -1, 128, 65, 65]	0
Conv2d-17	[ -1, 256, 65, 65]	295,168
BatchNorm2d-18	[ -1, 256, 65, 65]	512
LeakyReLU-19	[ -1, 256, 65, 65]	0
MaxPool2d-20	[ -1, 256, 32, 32]	0
Conv2d-21	[ -1, 256, 32, 32]	590,080
BatchNorm2d-22	[ -1, 256, 32, 32]	512
LeakyReLU-23	[ -1, 256, 32, 32]	0
ConvTranspose2d-24	[ -1, 256, 64, 64]	590,080
BatchNorm2d-25	[ -1, 256, 64, 64]	512
LeakyReLU-26	[ -1, 256, 64, 64]	0
ConvTranspose2d-27	[ -1, 128, 128, 128]	295,040
BatchNorm2d-28	[ -1, 128, 128, 128]	256

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
LeakyReLU-29	[ -1, 128, 128, 128 ]	0
Conv2d-30	[ -1, 128, 128, 128 ]	147,584
BatchNorm2d-31	[ -1, 128, 128, 128 ]	256
LeakyReLU-32	[ -1, 128, 128, 128 ]	0
Conv2d-33	[ -1, 64, 128, 128 ]	73,792
BatchNorm2d-34	[ -1, 64, 128, 128 ]	128
LeakyReLU-35	[ -1, 64, 128, 128 ]	0
Conv2d-36	[ -1, 32, 128, 128 ]	18,464
BatchNorm2d-37	[ -1, 32, 128, 128 ]	64
LeakyReLU-38	[ -1, 32, 128, 128 ]	0
Conv2d-39	[ -1, 16, 130, 130 ]	12,816
BatchNorm2d-40	[ -1, 16, 130, 130 ]	32
LeakyReLU-41	[ -1, 16, 130, 130 ]	0
Conv2d-42	[ -1, 3, 128, 128 ]	2,355
Tanh-43	[ -1, 3, 128, 128 ]	0

Visto la stazionarietà nei risultati delle 2 architetture precedenti, ho deciso di operare sulle dimensioni dei kernel.

In particolare, questa architettura si distingue dalle altre per l'utilizzo di kernel di dimensioni variabili nelle convoluzioni. Nello specifico, i primi due strati convoluzionali utilizzano kernel di dimensioni 7 e 5 rispettivamente, con padding adeguato per mantenere le dimensioni dell'immagine, con il goal di catturare caratteristiche più ampie e globali dell'immagine sin dalle prime fasi di codifica mentre, nel decoder, gli ultimi due strati convoluzionali invertiti utilizzano kernel di dimensioni 5 e 7 rispettivamente, rispetto ai kernel di dimensione 3 delle precedenti architetture, per favorire una ricostruzione più uniforme e ridurre gli artefatti.

In generale queste modifiche servono per migliorare la capacità del modello di apprendere rappresentazioni spaziali più complesse e di ottenere una migliore qualità nella generazione delle immagini ricostruite.

Output epochs:

```

Epoch 0: 100%|          13/13 [04:46<00:00,  0.05it/s, v_num=47, train/loss_step=0.0866,
      val/loss_step=0.124, val/loss_epoch=0.204, train/loss_epoch=0.217]
Metric val/loss improved. New best score: 0.204
Epoch 1: 100%|          13/13 [04:49<00:00,  0.04it/s, v_num=47, train/loss_step=0.0278,
      val/loss_step=0.045, val/loss_epoch=0.0741, train/loss_epoch=0.0505]
Metric val/loss improved by 0.129 >= min_delta = 0.005. New best score: 0.074
Epoch 2: 100%|          13/13 [04:48<00:00,  0.05it/s, v_num=47, train/loss_step=0.0256,
      val/loss_step=0.0234, val/loss_epoch=0.0269, train/loss_epoch=0.0249]
Metric val/loss improved by 0.047 >= min_delta = 0.005. New best score: 0.027
Epoch 3: 100%|          13/13 [04:48<00:00,  0.04it/s, v_num=47, train/loss_step=0.0273,
      val/loss_step=0.0212, val/loss_epoch=0.0197, train/loss_epoch=0.0205]
Metric val/loss improved by 0.007 >= min_delta = 0.005. New best score: 0.020
Epoch 13: 100%|         13/13 [04:49<00:00,  0.04it/s, v_num=47, train/loss_step=0.012,
      val/loss_step=0.0171, val/loss_epoch=0.0162, train/loss_epoch=0.0155]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.020. Signaling Trainer to stop.
Epoch 13: 100%|         13/13 [04:49<00:00,  0.04it/s, v_num=47, train/loss_step=0.012,
      val/loss_step=0.0171, val/loss_epoch=0.0162, train/loss_epoch=0.0155]

```

In questa architettura possiamo notare una capacità del modello di convergere molto più velocemente, tuttavia i risultati continuano a restare uguali, non portando dunque alcun miglioramento visivo nelle immagini ricostruire.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	17.78 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.4483

Table 5.19: Valori medi di PSNR e SSIM

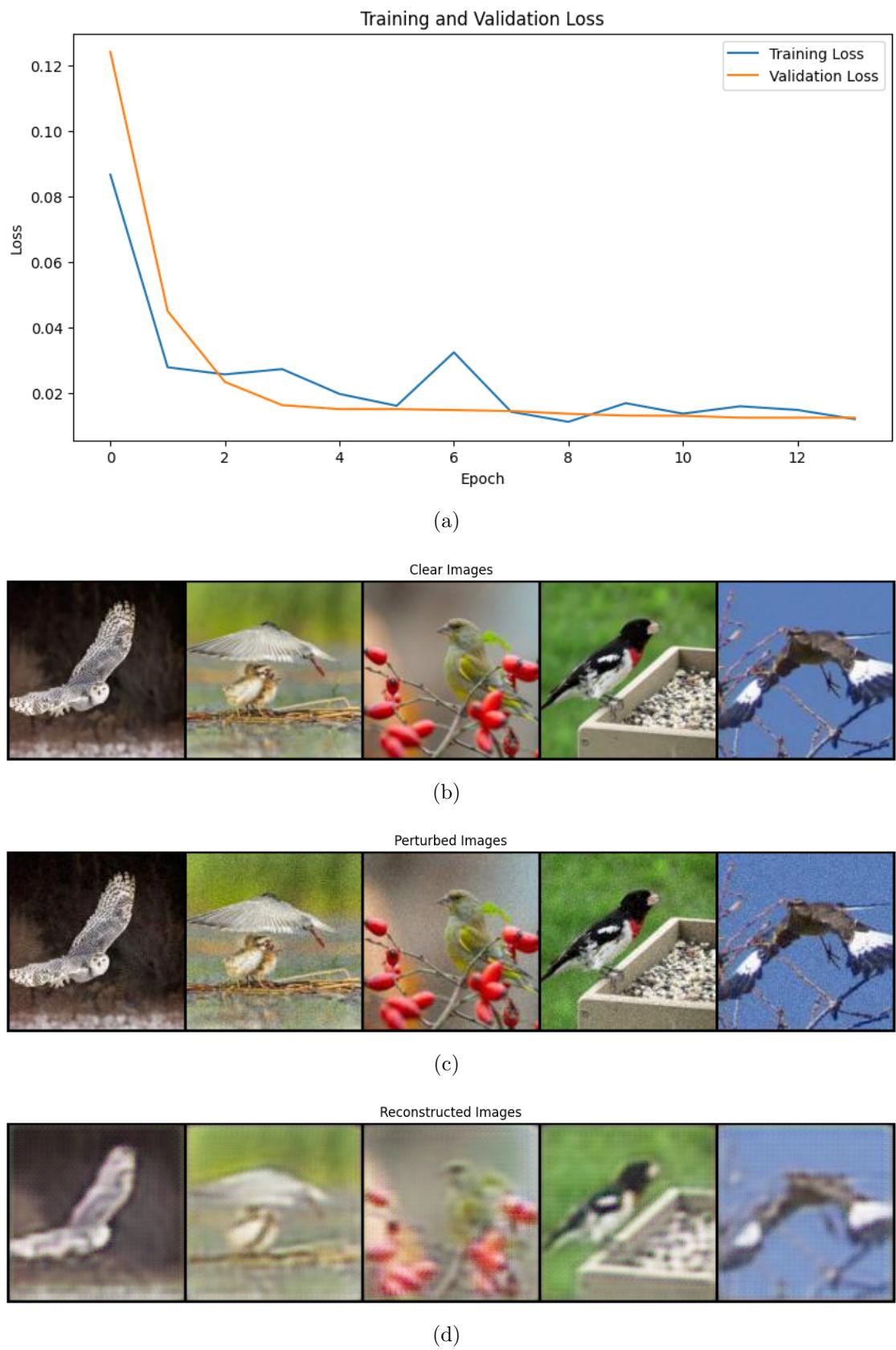


Figure 5.9: Risultati nono esperimento

## 5.10 Decimo esperimento

Table 5.20: Modello decimo esperimento

<b>Layer (type)</b>	<b>Output Shape</b>	<b>Param #</b>
Conv2d-1	[-1, 16, 128, 128]	2,368
BatchNorm2d-2	[-1, 16, 128, 128]	32
LeakyReLU-3	[-1, 16, 128, 128]	0
Conv2d-4	[-1, 32, 128, 128]	12,832
BatchNorm2d-5	[-1, 32, 128, 128]	64
LeakyReLU-6	[-1, 32, 128, 128]	0
MaxPool2d-7	[-1, 32, 64, 64]	0
Conv2d-8	[-1, 64, 64, 64]	18,496
BatchNorm2d-9	[-1, 64, 64, 64]	128
LeakyReLU-10	[-1, 64, 64, 64]	0
Conv2d-11	[-1, 128, 64, 64]	73,856
BatchNorm2d-12	[-1, 128, 64, 64]	256
LeakyReLU-13	[-1, 128, 64, 64]	0
Conv2d-14	[-1, 256, 64, 64]	295,168
BatchNorm2d-15	[-1, 256, 64, 64]	512
LeakyReLU-16	[-1, 256, 64, 64]	0
ConvTranspose2d-17	[-1, 128, 128, 128]	295,040
BatchNorm2d-18	[-1, 128, 128, 128]	256
LeakyReLU-19	[-1, 128, 128, 128]	0
Conv2d-20	[-1, 64, 128, 128]	73,792
BatchNorm2d-21	[-1, 64, 128, 128]	128
LeakyReLU-22	[-1, 64, 128, 128]	0
Conv2d-23	[-1, 32, 128, 128]	18,464
BatchNorm2d-24	[-1, 32, 128, 128]	64
LeakyReLU-25	[-1, 32, 128, 128]	0
Conv2d-26	[-1, 16, 130, 130]	12,816
BatchNorm2d-27	[-1, 16, 130, 130]	32
LeakyReLU-28	[-1, 16, 130, 130]	0
Conv2d-29	[-1, 3, 128, 128]	2,355
Tanh-30	[-1, 3, 128, 128]	0

In quest'architettura ho deciso di rendere l'autoencoder meno profondo, scendendo le immagini di input ad una dimensione di 64x64.

Output epochs:

```

Epoch 0: 100%|          13/13 [3:06:00<00:00,  0.00it/s, v_num=48,
      train/loss_step=0.0734, val/loss_step=0.126, val/loss_epoch=0.200, train/loss_epoch=0.181]
Metric val/loss improved. New best score: 0.200
Epoch 1: 100%|          13/13 [02:04<00:00,  0.10it/s, v_num=48, train/loss_step=0.0273,
      val/loss_step=0.0402, val/loss_epoch=0.0548, train/loss_epoch=0.0381]
Metric val/loss improved by 0.145 >= min_delta = 0.005. New best score: 0.055
Epoch 2: 100%|          13/13 [01:59<00:00,  0.11it/s, v_num=48, train/loss_step=0.0104,
      val/loss_step=0.0239, val/loss_epoch=0.0215, train/loss_epoch=0.0207]
Metric val/loss improved by 0.033 >= min_delta = 0.005. New best score: 0.021
Epoch 6: 100%|          13/13 [01:59<00:00,  0.11it/s, v_num=48,
      train/loss_step=0.00556, val/loss_step=0.0186, val/loss_epoch=0.0165, train/loss_epoch=0.0161]
Metric val/loss improved by 0.005 >= min_delta = 0.005. New best score: 0.016
Epoch 16: 100%|         13/13 [01:59<00:00,  0.11it/s, v_num=48,
      train/loss_step=0.0151, val/loss_step=0.0158, val/loss_epoch=0.0143, train/loss_epoch=0.0143]
Monitored metric val/loss did not improve in the last 10 records. Best score: 0.016. Signaling Trainer to stop.
Epoch 16: 100%|         13/13 [01:59<00:00,  0.11it/s, v_num=48,
      train/loss_step=0.0151, val/loss_step=0.0158, val/loss_epoch=0.0143, train/loss_epoch=0.0143]
```

La piccola modifica fatta ha portato ad un leggero miglioramento visivo, in particolare in un miglioramento della SSIM, rispetto alla precedente architettura, da 0.4483 a 0.4980.

Descrizione	Valore
Average PSNR between clear and perturbed images	30.73 dB
Average PSNR between clear and reconstructed images	16.85 dB
Average SSIM between clear and perturbed images	0.7557
Average SSIM between clear and reconstructed images	0.4980

Table 5.21: Valori medi di PSNR e SSIM

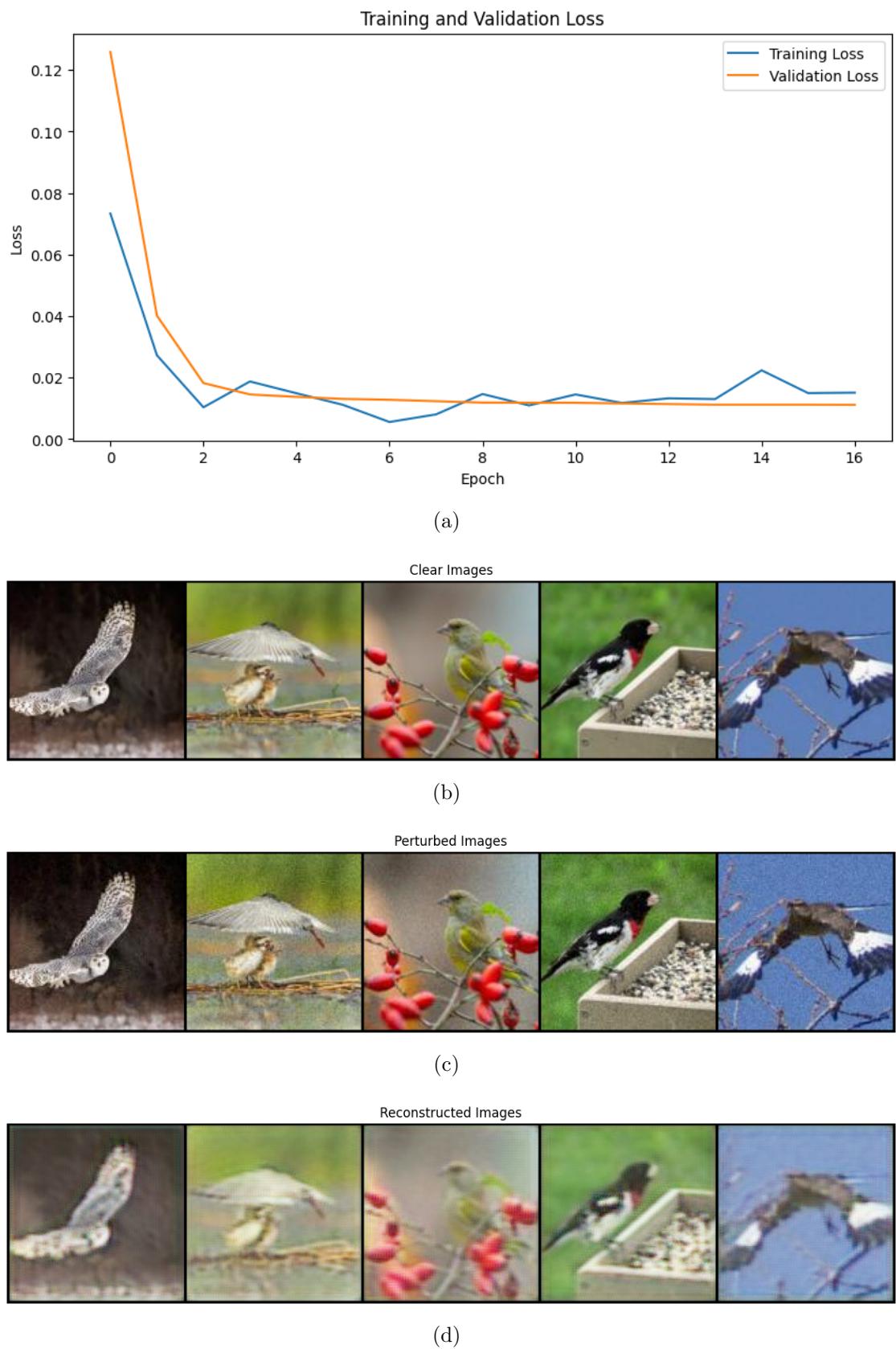


Figure 5.10: Risultati decimo esperimento

# Chapter 6

## Demo

La demo proposta è disponibile nell'ultima sezione del notebook esaminato nel Cap.7.

All'interno del notebook viene caricato in memoria il modello che in fase di sperimentazione ha dato i risultati migliori ed il suo stato salvato nella cartella "models" grazie alla funzione `load_state_dict()` della libreria pytorch.



(a)



(b)

Figure 6.1: Immagini demo senza rumore

Successivamente vengono caricate le immagini presenti all'interno della cartella "demo" e scalate, grazie alla funzione resize di skimage, a 128x128 (Fig.6.1).

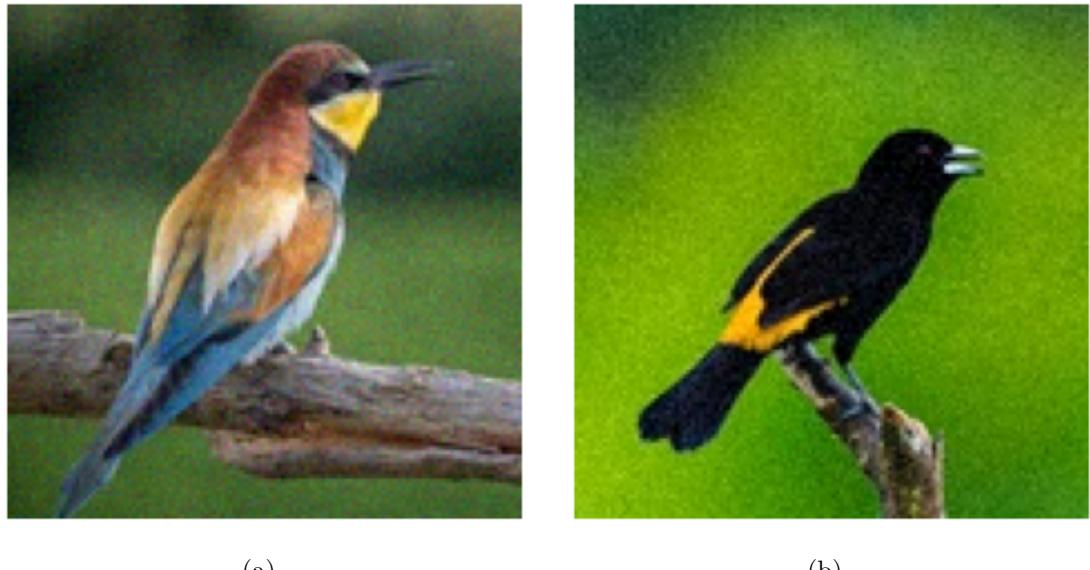


Figure 6.2: Immagini demo con rumore

Dopodichè viene aggiunto rumore cromatico alla prima e luminoso alla seconda (Fig.6.2)

Infine viene caricato il modello col suo stato ed usato per rimuovere il rumore dalle immagini caricate.

Come ci si poteva aspettare dagli scarsi risultati durante la fase di sperimentazione, le immagini ricostruite senza rumore (Fig.6.3) sono molto lontane da quelle originali (Fig.6.1).

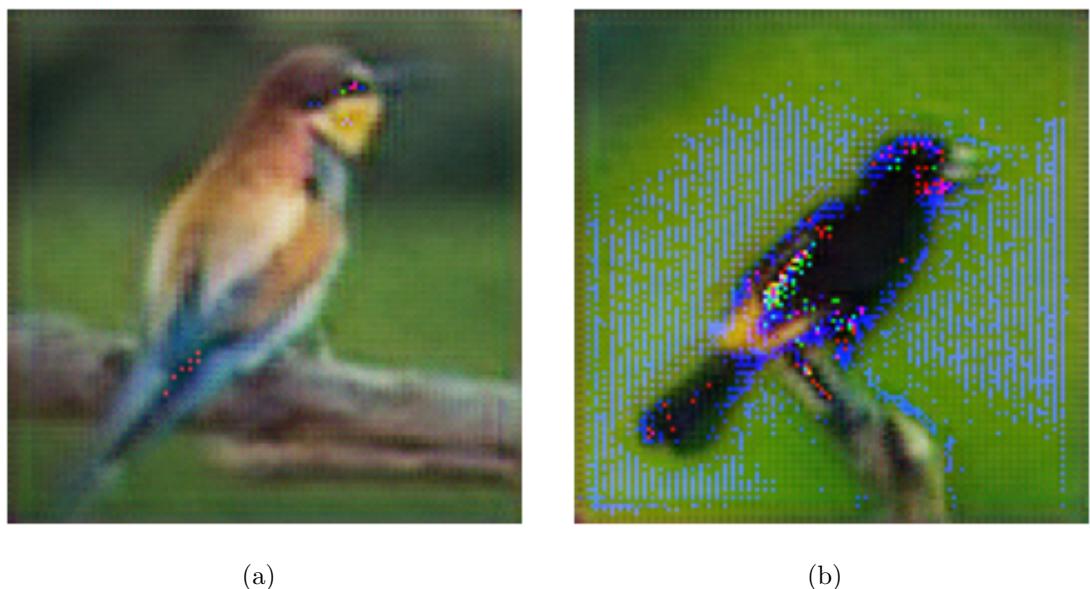


Figure 6.3: Immagini demo ricostruite senza rumore

# Chapter 7

## Codice

In questo capitolo non mostrerò il codice completo ma mostrerò com'è organizzato il notebook ed il funzionamento.

Il notebook completo resta comunque disponibile alla seguente repository Rep.[3]

### 7.1 Notebook Denoising\_Autoencoder.ipynb

Il notebook Denoising\_Autoencoder.ipynb contiene tutti gli esperimenti fatti, con relativi modelli e risultati, e alcune funzioni e procedure utili per il preprocessing del dataset e calcolo di metriche e plot ai fini della valutazione.

#### 7.1.1 Preprocessing immagini

La parte di preprocessing contiene importanti fasi della creazione del dataset.

In ordine:

- **Cropping, resizing & renaming:** in questa fase le immagini vengono ritagliata e ridimensionate a 128x128 per poi infine essere numerate e salvate nella cartella "dataset"
- **Aggiunta rumore:** l'aggiunta dei rumori viene fatta tramite 2 funzioni chiamate "add\_chromatic\_noise" e "add\_luminance\_noise" che, prendendo in input l'immagine ed il livello di noise desiderato, aggiungono il rispetto rumore

seguendo una distribuzione gaussiana. A fine procedura si ottengono le cartelle "dataset\_chromatic\_noise" e "dataset\_luminance\_noise".

### 7.1.2 Utility functions

Qui ho definito una serie di funzioni utilizzate durante la fase di sperimentazione.

- **calculate\_psnr & calculate\_ssim**: due funzioni utili per il calcolo delle due metriche
- **show\_images**: usata per mostrare, a fine training, le prime 5 immagini dell'ultima fase di validation del modello, mostrando inoltre anche le metriche PSNR e SSIM
- **NoisyImageDataset**: questa è una classe molto utile che permette di caricare in batches con il DataLoader offerto da pytorch le coppie di immagini originali con la loro versione con rumore.

### 7.1.3 Esperimenti

In questa sessione è possibile trovare tutti gli esperimenti descritti in Cap.5.

Tutte le architetture proposte sono state implementate sfruttando Pytorch Lightning al fine di averle più strutturate ed organizzate.

Ogni architettura è stata implementata definendo una lista contenente i risultati dell'ultima fase di validation ed estendendo le seguenti funzioni fornite dal framework:

- **training\_step & validation\_step**: queste funzioni le ho adattate per usare batch contenenti sia l'immagine originale che con rumore, entrambe utili per calcolare la MSE loss
- **on\_train\_epoch\_end & on\_validation\_epoch\_end**: ho sfruttato queste due funzioni per conservare le loss ottenute

- **on\_train\_start**: questa parte l'ho cambiata per stampare una summary più dettagliata dell'architettura
- **on\_train\_end**: qui ho aggiunto la fase di stampa dei plot di train e val loss nonchè il salvataggio dello stato del modello

#### 7.1.4 Demo

Quest'ultima sezione contiene la demo del progetto dove viene mostrata l'efficacia del miglior modello ottenuto con 2 immagini non presenti nel dataset di training e validation.

# Chapter 8

## Conclusioni

Il progetto proposto ha comportato diverse fasi, dalle operazioni di acquisizione e pulizia dei dati fino all’addestramento e valutazione del modello.

Nonostante gli sforzi i risultati ottenuti non sono stati quelli sperati, tuttavia il processo di sviluppo del progetto mi ha permesso di cogliere alcune osservazioni interessanti.

### 8.1 Considerazioni riguardanti il dataset

Ho acquisito il dataset tramite una semplice ricerca su google immagini.

Questa fase ha portato a un dataset contenente diverse immagini, seguite da operazioni di pulizia per rimuovere immagini di scarsa qualità o con watermark ed operazioni di cropping.

Nonostante questi sforzi, il dataset rimane comunque limitato in termini di dimensione, il che, visti i risultati e tentativi vari, ha sicuramente rappresentato essere un fattore limitante.

### 8.2 Considerazioni riguardanti gli esperimenti

Durante lo sviluppo del progetto, sono stati creati e testati diversi modelli di deep learning utilizzando come esempio gli autoencoder.

Sono stati effettuati diversi esperimenti con architetture diverse, introducendo strati di convoluzione, pooling, batch normalization.

Tuttavia, nonostante gli sforzi per migliorare i modelli, i risultati non sono stati estremamente soddisfacenti. Questo può essere attribuito alla complessità del problema e alle limitazioni del dataset.

Sono stati utilizzati diversi strumenti per la valutazione dei modelli, come la MSE loss ed i grafici di train e validation loss nonché metriche molto diffuse in computer vision quali PSNR e SSIM.

Questi strumenti hanno aiutato a capire come il modello si sia comportato durante il training evidenziando la difficoltà di ricostruire immagini senza rumore.

### 8.3 Lezioni apprese

Lo sviluppo del progetto e la conseguente stesura della relazione mi ha fornito diverse lezioni:

1. **Importanza di un dataset di dimensioni adeguate:** È cruciale disporre di un dataset abbastanza ampio per addestrare un modello in modo efficace. Dati limitati possono portare il modello a non avere abbastanza esempi per cogliere le caratteristiche intrinseche del problema affrontato.
2. **Regolazione dei modelli:** Ho provato ad affrontare il problema con diverse architetture, modificandone la complessità e aggiungendo o rimuovendo strati. Da questo ho sicuramente appreso che più complesso non implica necessariamente un miglioramento delle prestazioni. Spesso, infatti, i migliori risultati li ho ottenuti con architetture più semplici.
3. **Importanza delle metriche:** Ho sicuramente apprezzato l'importanza delle metriche per valutare correttamente le prestazioni del modello. In particolare

la SSIM mi ha permesso di capire quanto distante fossero, a livello visivo e percettivo, le immagini ricostruite da quelle originali

4. **Hardware:** Ho anche appreso l'importanza di avere delle risorse computazionali potenti per poter affrontare al meglio questa tipologia di problemi. Infatti la scarsità di risorse mi ha limitato nella fase di sperimentazione a causa delle lunghe attese dovute alle fasi di training.

## 8.4 Sviluppi futuri

Ho diversi spunti di miglioramento:

1. **Aumentare la quantità di dati:** Un dataset più grande e vario potrebbe aiutare il modello a cogliere meglio le caratteristiche dei dati e migliorare le prestazioni.
2. **Tuning degli iperparametri:** Esplorare una gamma più ampia di iperparametri potrebbe aiutare a trovare una configurazione migliore per il modello.

# Bibliography

- [1] L. Gondara. “Medical Image Denoising Using Convolutional Denoising Autoencoders”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 2016, pp. 241–246. DOI: [10.1109/ICDMW.2016.0041](https://doi.org/10.1109/ICDMW.2016.0041).
- [2] K. Bajaj, D. K. Singh, and M. A. Ansari. “Autoencoders Based Deep Learner for Image Denoising”. In: *Procedia Computer Science* 171 (2020). Third International Conference on Computing and Network Communications (CoCoNet’19), pp. 1535–1541. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2020.04.164>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050920311431>.
- [3] G. Condorelli. *Deep Learning Project Repository 2023-2024*. <https://github.com/Gpp23/Denoising-autoencoder>.