A Algorithms for Generating Fractal Images

In this section, we demonstrate the details of how we generate the labeled fractal images. Algorithm 1 shows how we use the escape algorithm to generate Julia Set. Algorithm 2 illustrates how to generate fractal images based on given grammar and other parameters of L-systems.

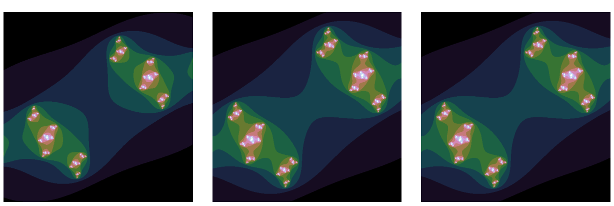
Algorithm 1 Julia Set Generation Algorithm

Require: Image width w, Image height h, Constant c, Iteration number N, scaling coefficient τ

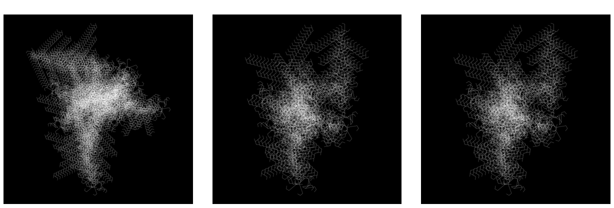
Julia set = 2D array with dimensions $w \times h$
for each pixel $(x, y) \in I$ **do**
  Set the complex number $z = \frac{x}{\tau} + \frac{y}{\tau}j$
  $f(z) = z^2 + c$
  for $i \in (1, N)$ **do**
    $z = f(z)$
    if $\|z\| > float('inf')$ **then**
      $value = i$
      break
    end if
    $value = i$
  end for
  $I(x, y) = value$
end for

B More Visualization Results

We provide additional visualization results in this section. Figure 8 demonstrates the results when employing different learning paradigms under the Densenet architecture. Figure 9 and 10 showcase the visualization results of the reconstruction fractal images using different encoder architecture in learning by min $L_1 + L_2 + L_3$ paradigms. Figure 11 and Figure 12 represent the results of different models and learning paradigms on the Julia Set and L-system-based fractal images.



(a) $L_1 + L_2$ (b) $L_1 + L_2 + L_3$ (c) Ground Truth



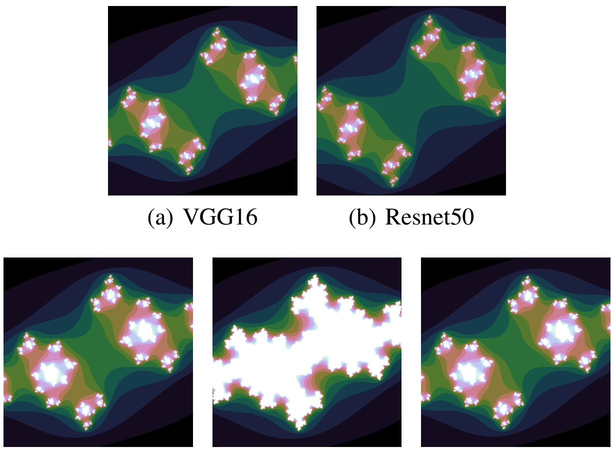
(d) $L_1 + L_2$ (e) $L_1 + L_2 + L_3$ (f) Ground Truth

Figure 8: Visual comparisons for Densenet by different learning paradigms.

Algorithm 2 L-system Based Fractal Image Generation Algorithm

Require: Rule Dict "rule", Iteration number N, Forward direction "angle"

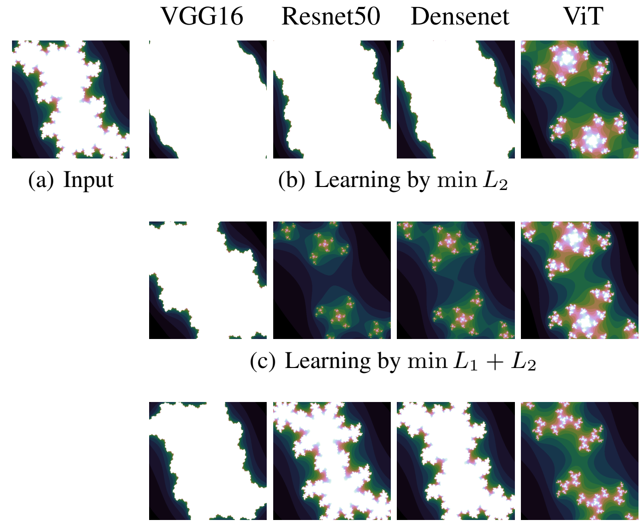
Fractal Image = a list of points composed of the image.
info = rule["S"]
for $i \in$ range(N) **do**
  temp = []
  for $c \in$ info **do**
    if $c \in$ rule **then**
      temp.append(rule[c])
    else
      temp.append(c)
    end if
  end for
  info = "".join(temp)
end for
lines, stack, d, p, l = [], [], 0, (0.0, 0.0), 1
for $c \in$ info **do**
  if $c =$ "F" **then**
    $r = (d) \mod 360 \times$ math.pi$/180$
    $t = (p[0] + l \times$ math.cos$(r), p[1] + l \times$ math.sin$(r))$
    lines.append$(((p[0], p[1]), (t[0], t[1])))$
    $p = t$
  else if $c ==$ " $+$ " **then**
    $d = d +$ angle
  else if $c ==$ " $-$ " **then**
    $d = d -$ angle
  else if $c ==$ "[" **then**
    stack.append$((p, d))$
  else if $c ==$ "]" **then**
    $p, d =$ stack$[-1]$
    del stack$[-1]$
  else if $c ==$ "!" **then**
    angle $=$ angle $\times (-1)$
  else if $c ==$ "|" **then**
    $d = d + 180$
  end if
end for
return lines



(a) VGG16 (b) Resnet50

(c) DenseNet (d) ViT (e) Ground Truth

Figure 9: Visual comparisons for various methods trained on the Julia Set.

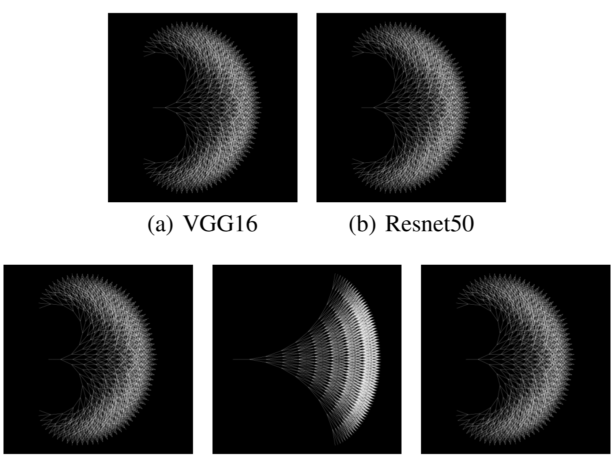


(a) Input (b) Learning by $\min L_2$

(c) Learning by $\min L_1 + L_2$

(d) Learning by $\min L_1 + L_2 + L_3$

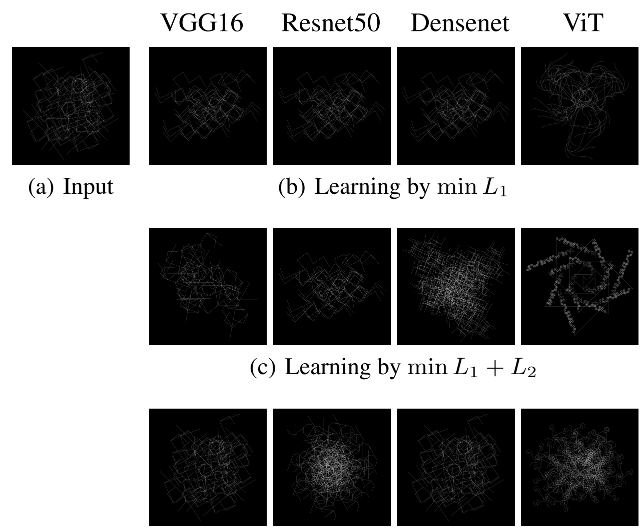
Figure 11: Visual comparisons for different models and learning paradigms given a Julia Set.



(a) VGG16 (b) Resnet50

(c) DenseNet (d) ViT (e) Ground Truth

Figure 10: Visual comparisons for various methods trained on L-system fractal images.



(a) Input (b) Learning by $\min L_1$

(c) Learning by $\min L_1 + L_2$

(d) Learning by $\min L_1 + L_2 + L_3$

Figure 12: Visual comparisons for different models and learning paradigms given an L-system fractal image.