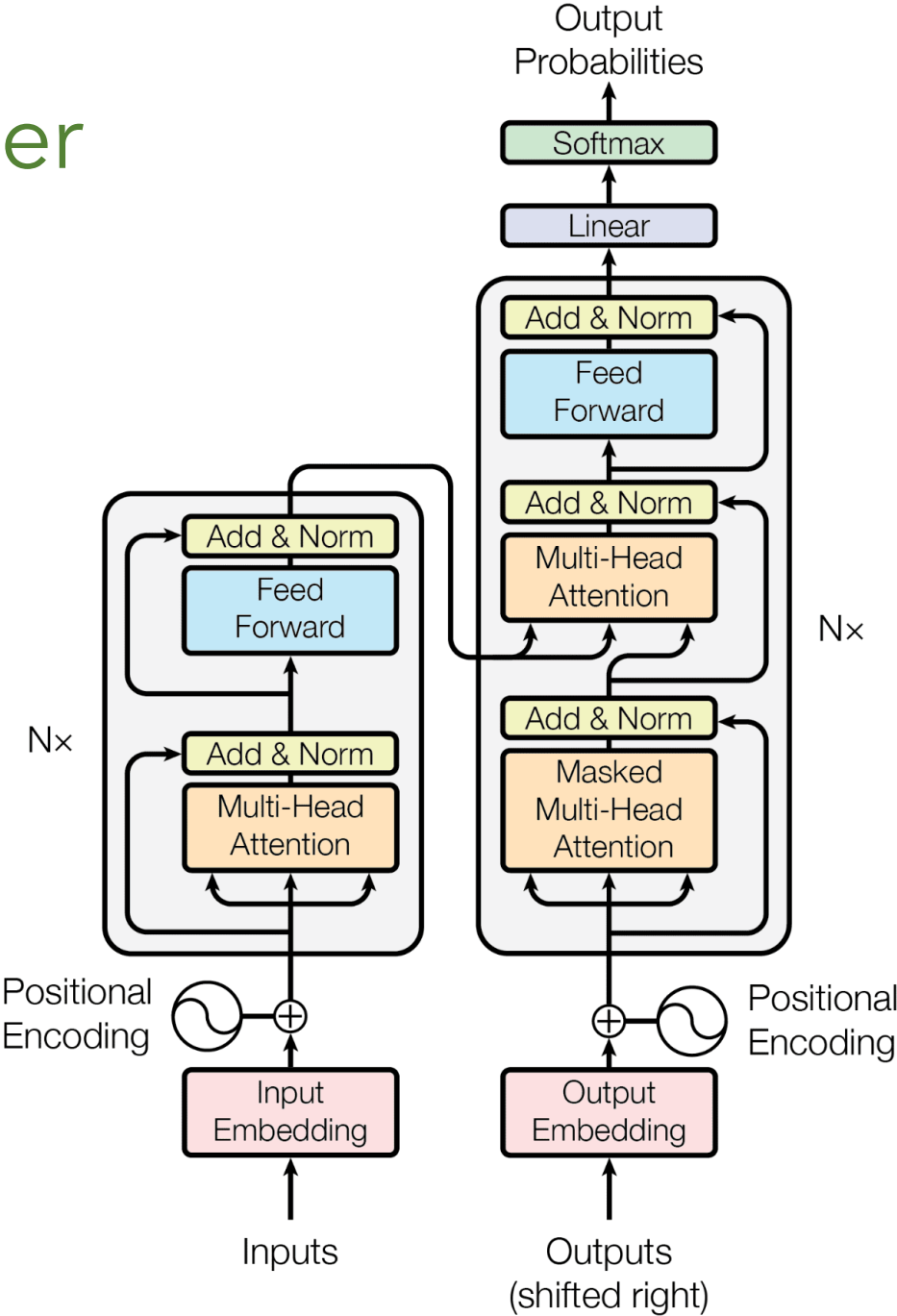
A futuristic garden scene with a maze-like path of glowing rectangular platforms. The garden is filled with various plants, including pink and purple flowers, and several stylized trees with glowing wireframe structures. The background features a grid of vertical lines and floating wireframe spheres and structures, creating a high-tech, digital atmosphere.

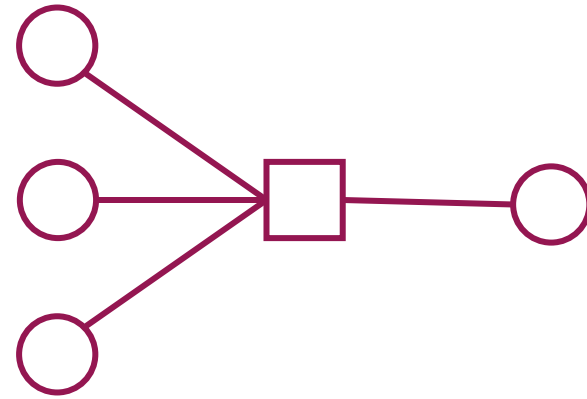
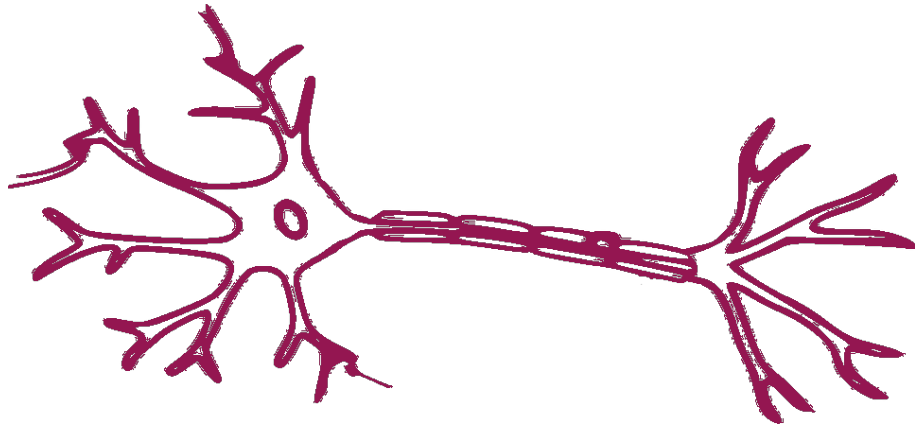
The
Evolution
of the
Transformer

The Transformer



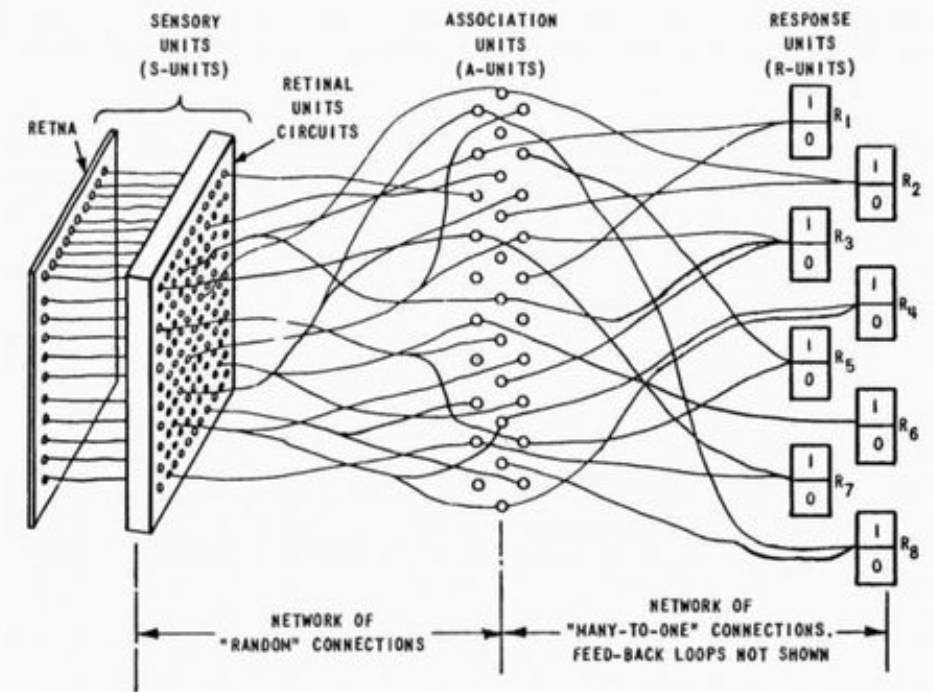
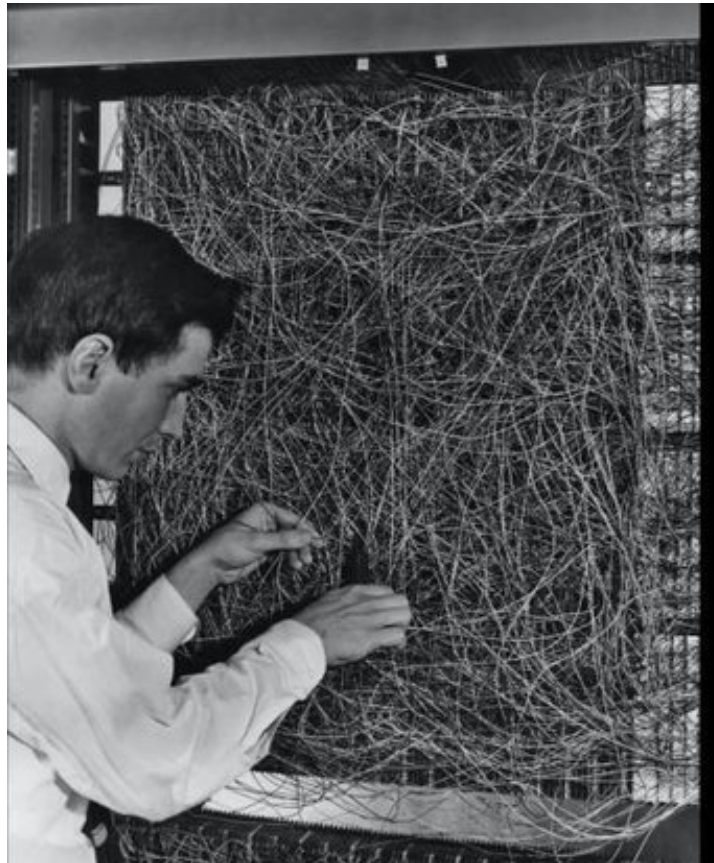
1943: The "Artificial Neuron"

Warren McCulloch and Walter Pitts



1957: The Perceptron

Frank Rosenblatt



1943:
The "Artificial Neuron"



1969: Perceptrons

Marvin Minsky and Seymour Papert

1957: The "Perceptron"

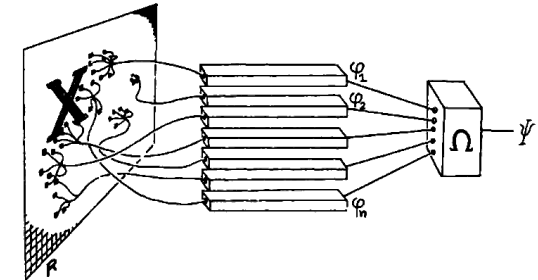
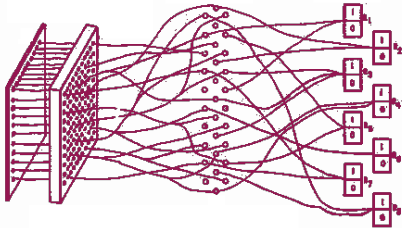


Figure 0.1

1970: Backpropagation

Seppo Linnainmaa

1943:
The "Artificial Neuron"



TAYLOR EXPANSION OF THE ACCUMULATED ROUNDING ERROR

- T1.* [Initialize.] $p \leftarrow n$, $C[i] \leftarrow 0$ for $i=1, \dots, m$, $i \neq N$, $C[N] \leftarrow 1$.
- T2.* [Read.] $i \leftarrow I[p]$, $j \leftarrow J[p]$, $k \leftarrow K[p]$, $dj \leftarrow Dj[p]$, $dk \leftarrow Dk[p]$.
- T3.* [Coefficient $c_{n,p}$ completed.] $\text{coef} \leftarrow C[i]$, $C[i] \leftarrow 0$. The contents of coef is now equal to the value of $c_{n,p}$ and can be utilized.
- T4.* [Coefficient zero?] If $\text{coef} = 0$, go to step *T6*.
- T5.* [Update table C .] If $dj \neq 0$ then $C[j] \leftarrow C[j] + \text{coef} \times dj$. If $dk \neq 0$ then $C[k] \leftarrow C[k] + \text{coef} \times dk$.
- T6.* [Decrease p .] Decrease p by 1. If $p > 0$, return to step *T2*, otherwise the algorithm terminates.

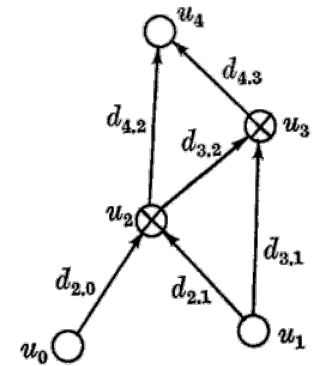
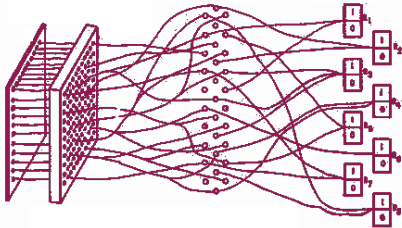
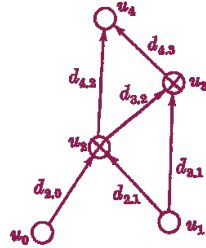


Figure 1. A computing process as a graph.

1957: The "Perceptron"



1970:
Backprop
proposed

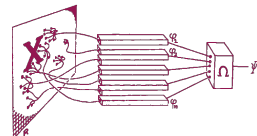


1970s & 1980s:
"AI Winter"

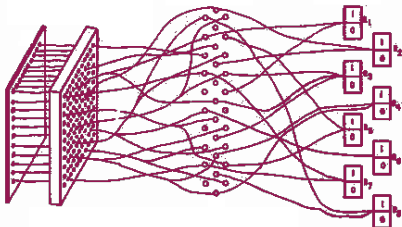
1943:
The "Artificial Neuron"



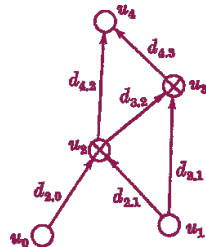
1969:
Perceptrons



1957: The "Perceptron"



1970:
Backprop
proposed

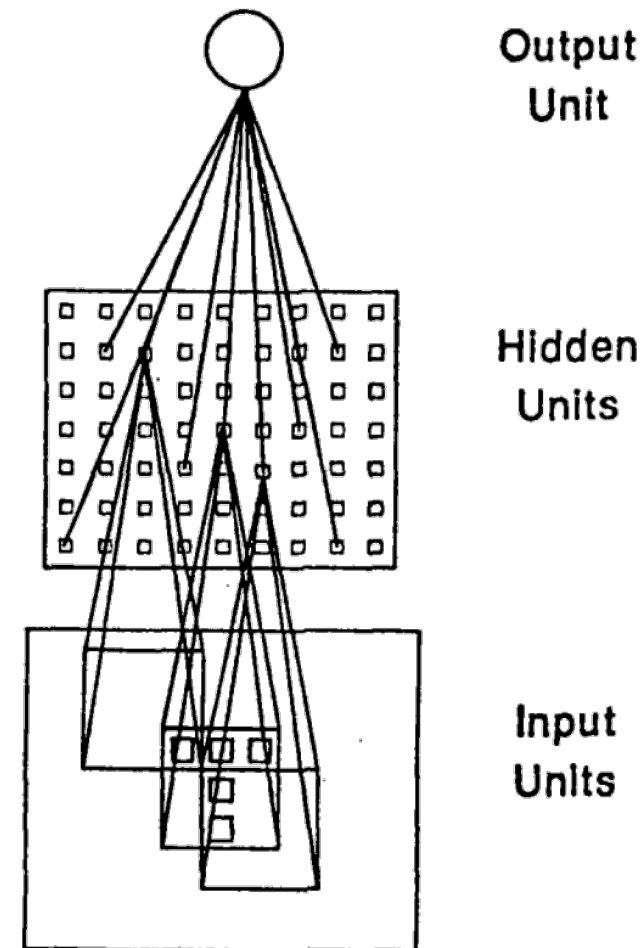
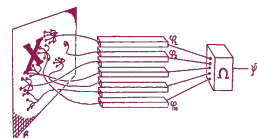


1986: Applied backprop
D Rumelhart, G Hinton, R Williams

1943:
The "Artificial Neuron"

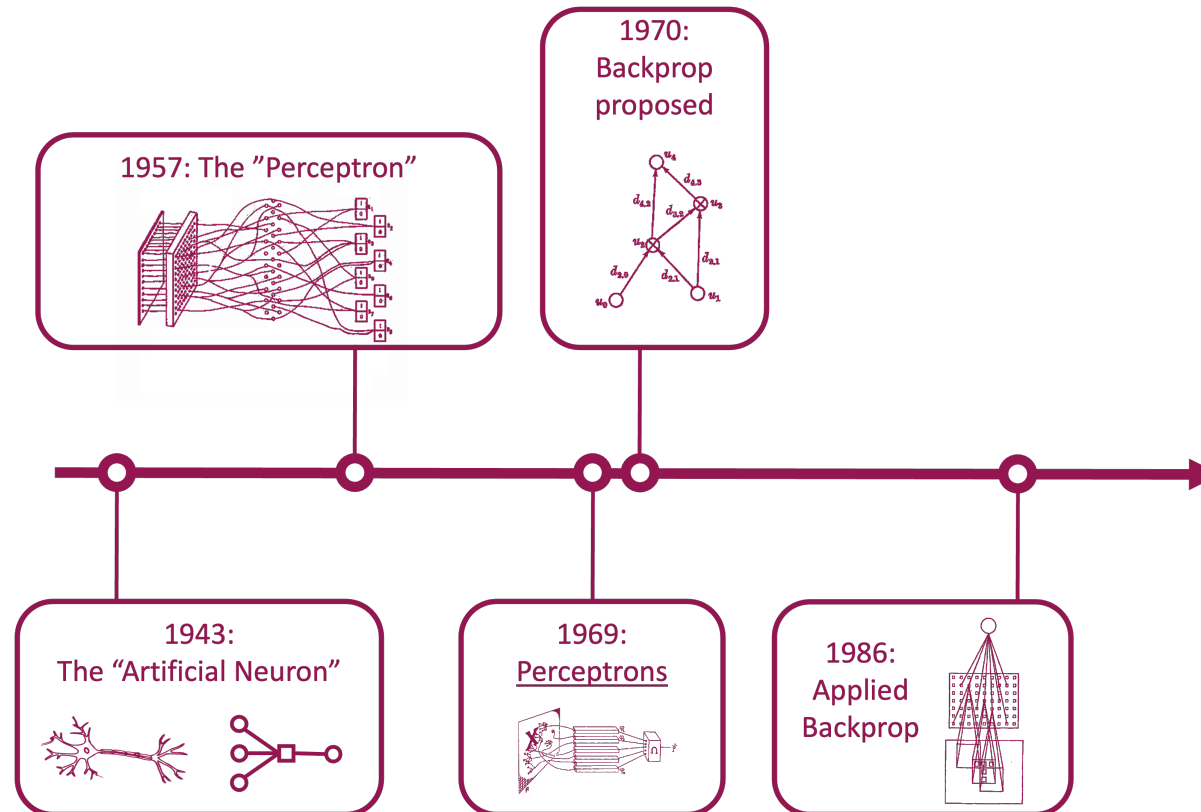


1969:
Perceptrons



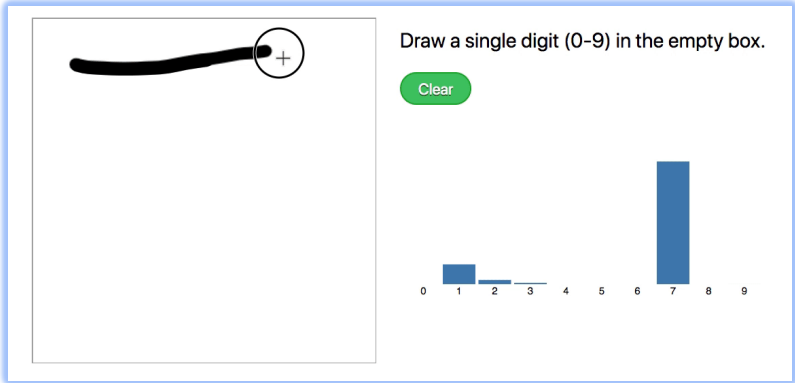
Neural Networking in the 20th Century

- **Linear & Nonlinear:** Composition of layers for computation
- **Trainable:** backpropagation



Computer Vision

A multitude of interesting challenges

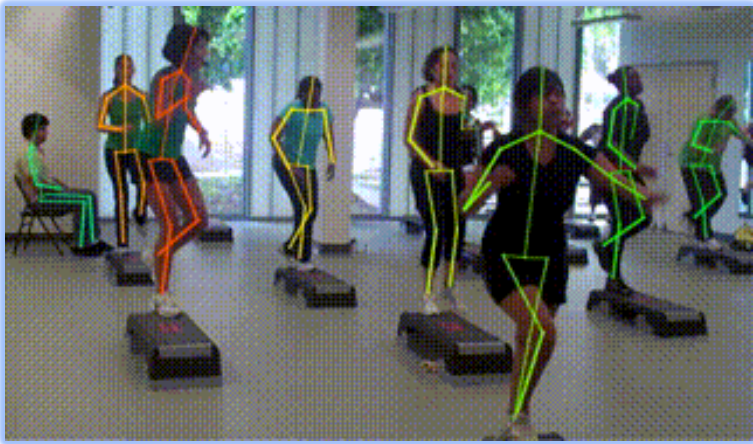
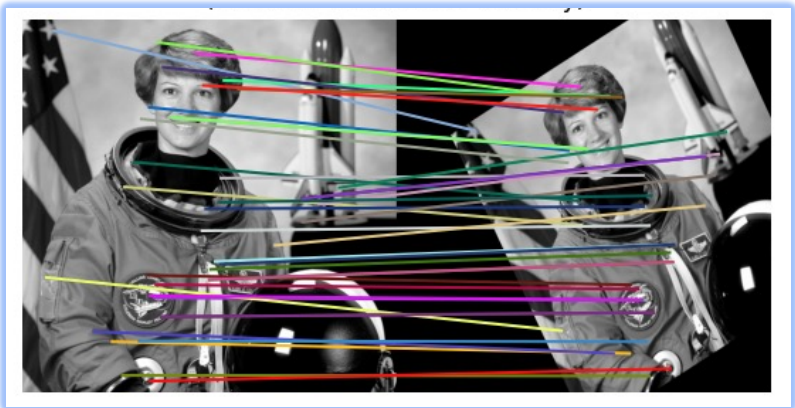
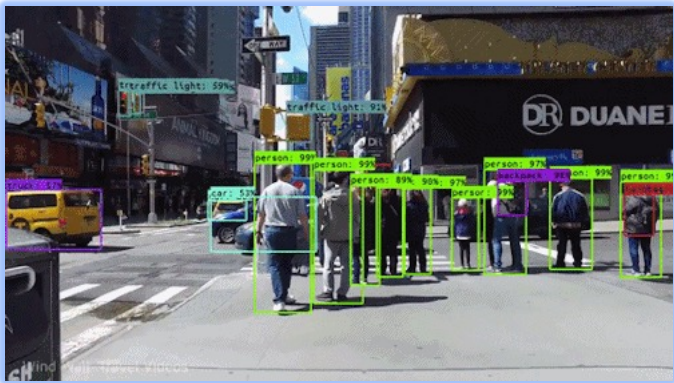


Draw a single digit (0-9) in the empty box.

Clear

Digit	Count
0	0
1	1
2	1
3	1
4	1
5	1
6	1
7	10
8	1
9	1

The image shows a digit recognition interface. On the left, there is a drawing area with a black scribble and a plus sign in a circle. On the right, there is a text prompt, a 'Clear' button, and a bar chart showing the frequency of digits drawn. The digit '7' has the highest frequency, with 10 occurrences.



ImageNet & ILSVRC (2009)

1,000 image categories

1.2 million images for training

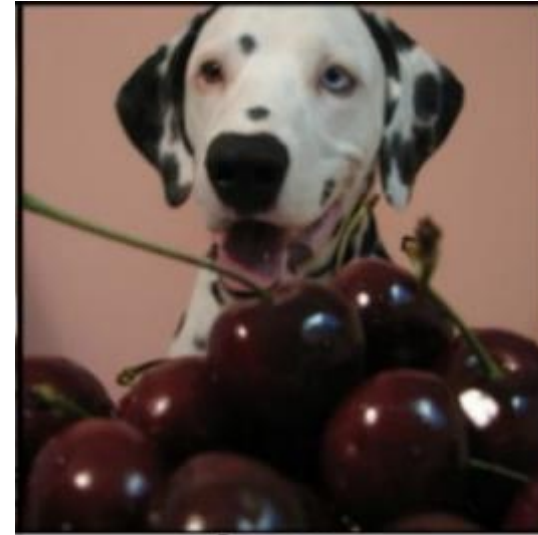
100k images for evaluation



container ship



mushroom

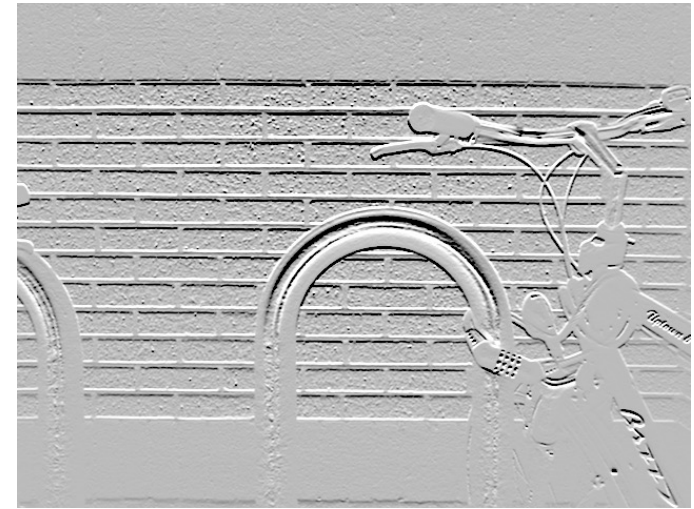
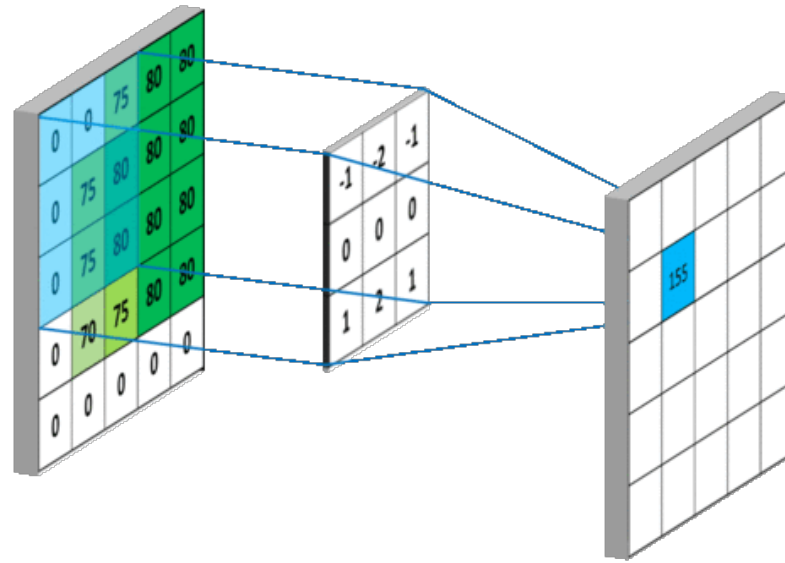


cherry

Filters

1	2	1
0	0	0
-1	-2	-1

Horizontal
Edge Detector

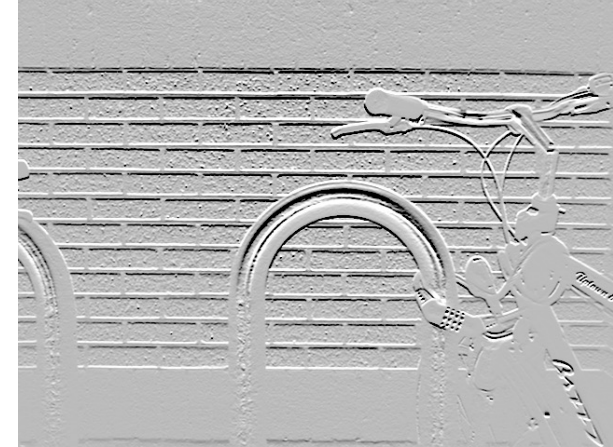


Filters



1	2	1
0	0	0
-1	-2	-1

Horizontal
Edge Detector



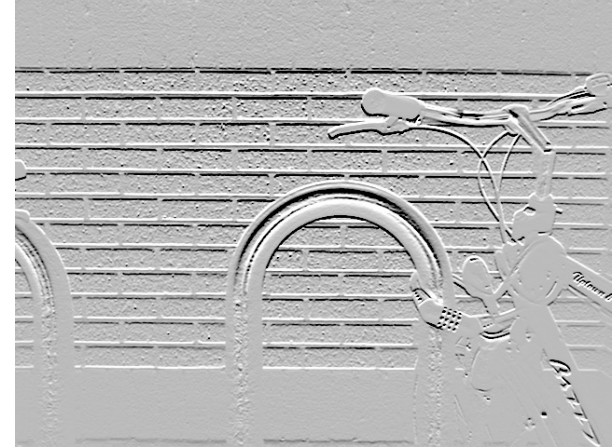
```
1 Gy = np.array([[1, 2, 1],
2                 [0, 0, 0],
3                 [-1, -2, -1]])
4 rows, columns = np.shape( grayscale_image )
5 sobel_filtered_image = np.zeros( shape=(rows, columns) )
6
7 for i in range( rows - 2 ):
8     for j in range( columns - 2 ):
9         gy = np.sum( np.multiply( Gy, grayscale_image[i:i+3, j:j+3] ) )
10        sobel_filtered_image[i + 1, j + 1] = gy
```

Filters



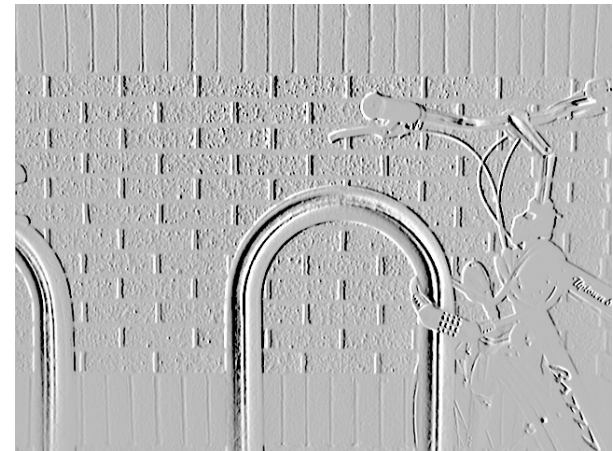
1	2	1
0	0	0
-1	-2	-1

Horizontal
Edge Detector

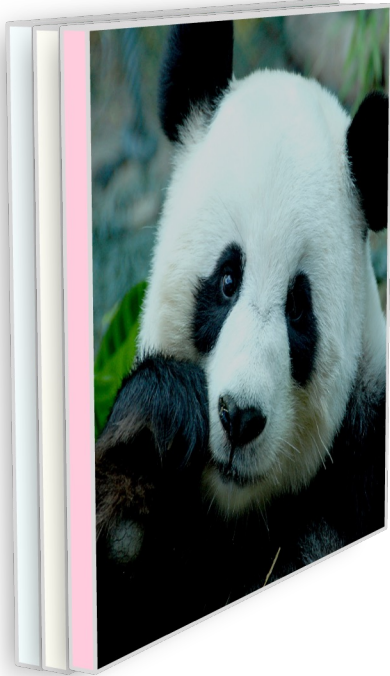


1	0	-1
2	0	-2
1	0	-1

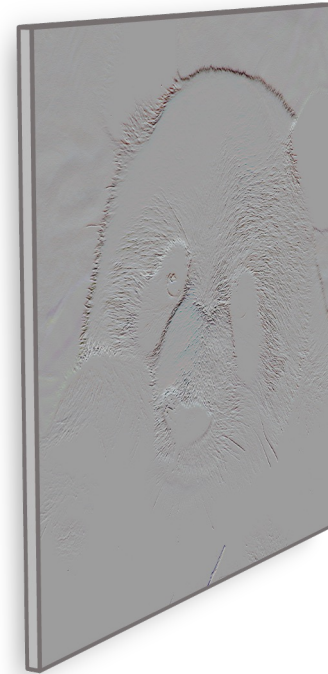
Vertical
Edge Detector



Filters



1	1	1	2	1
0	0	0	0	0
-1	-1	-1	-2	-1



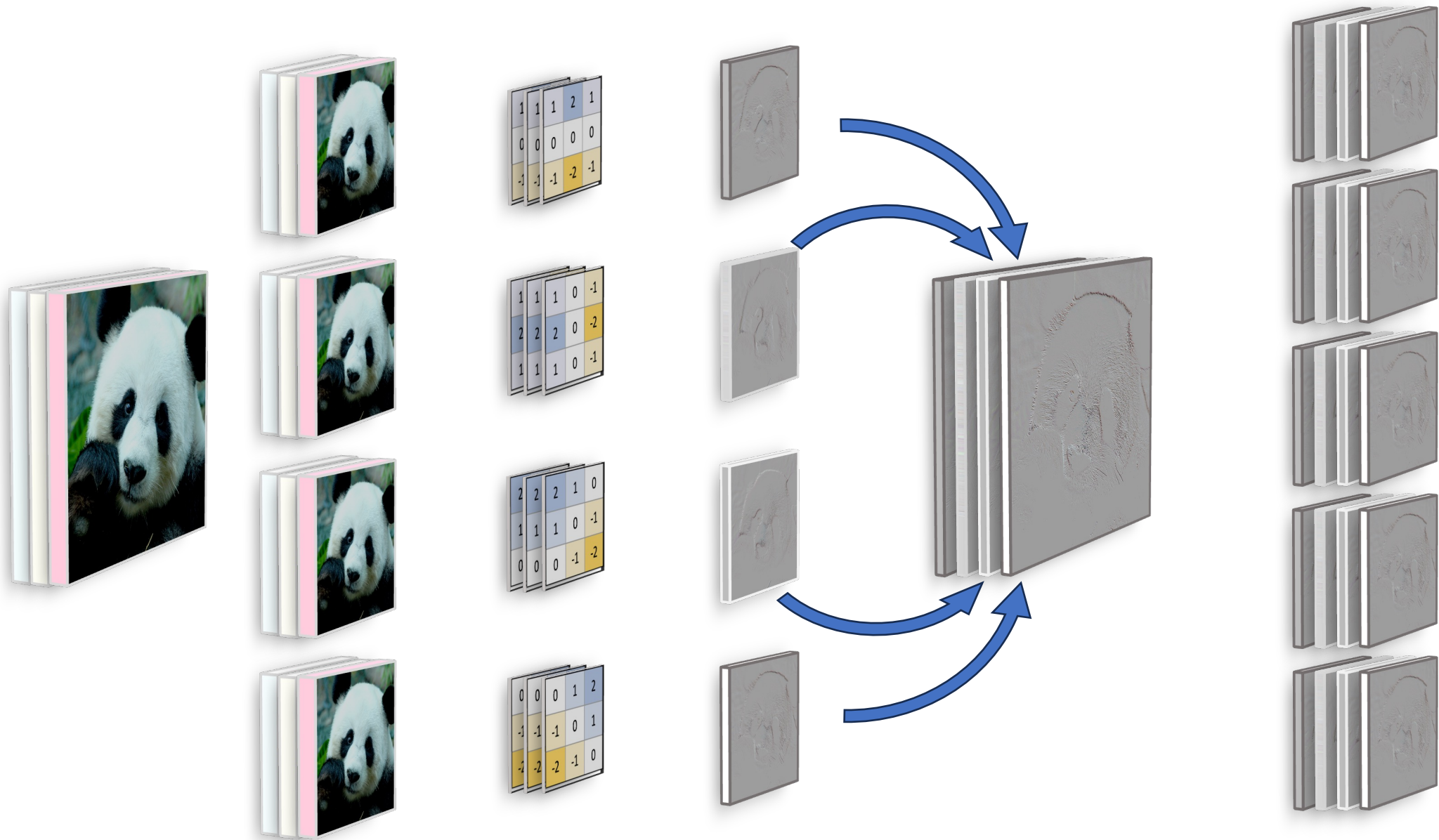
Filters



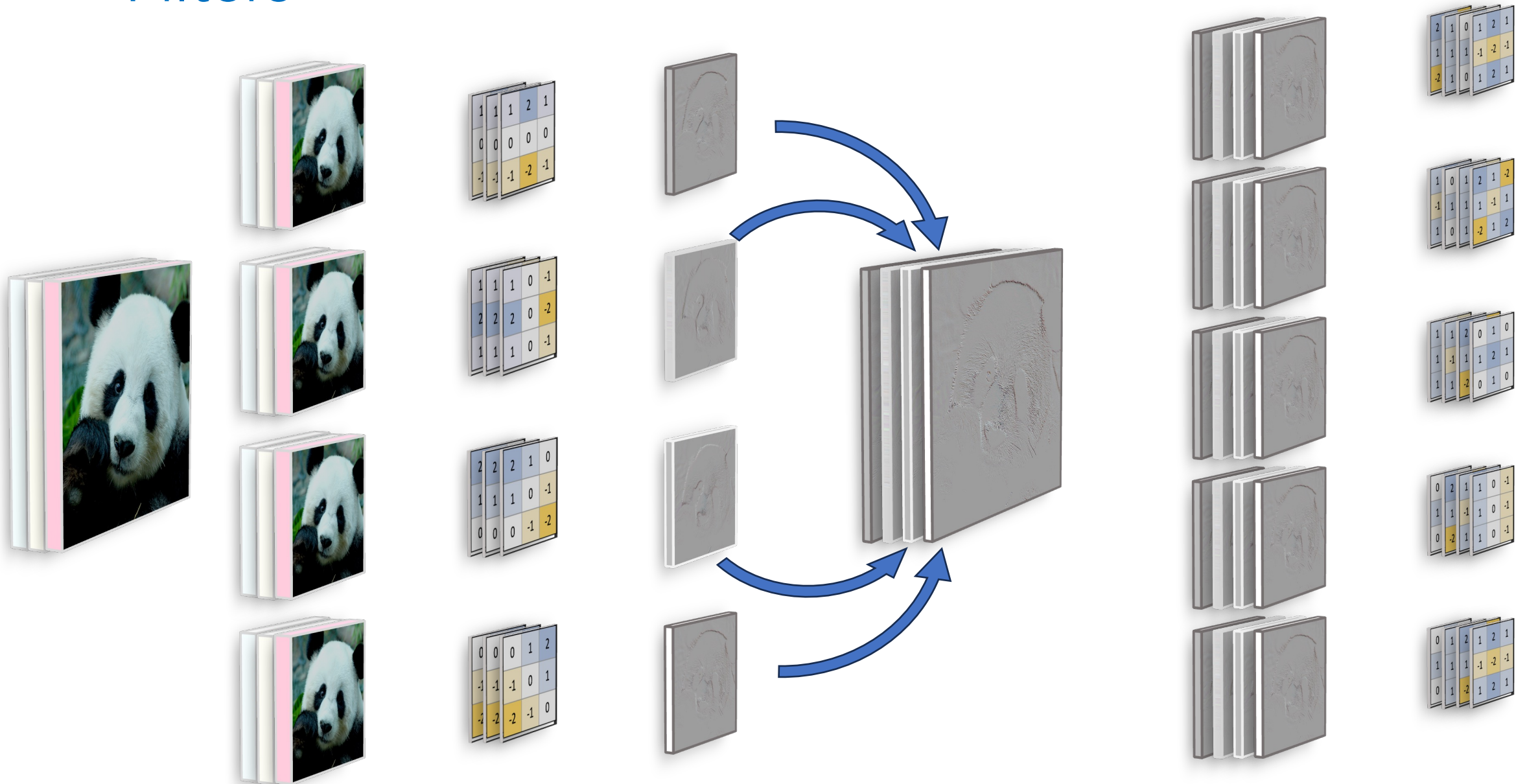
Filters



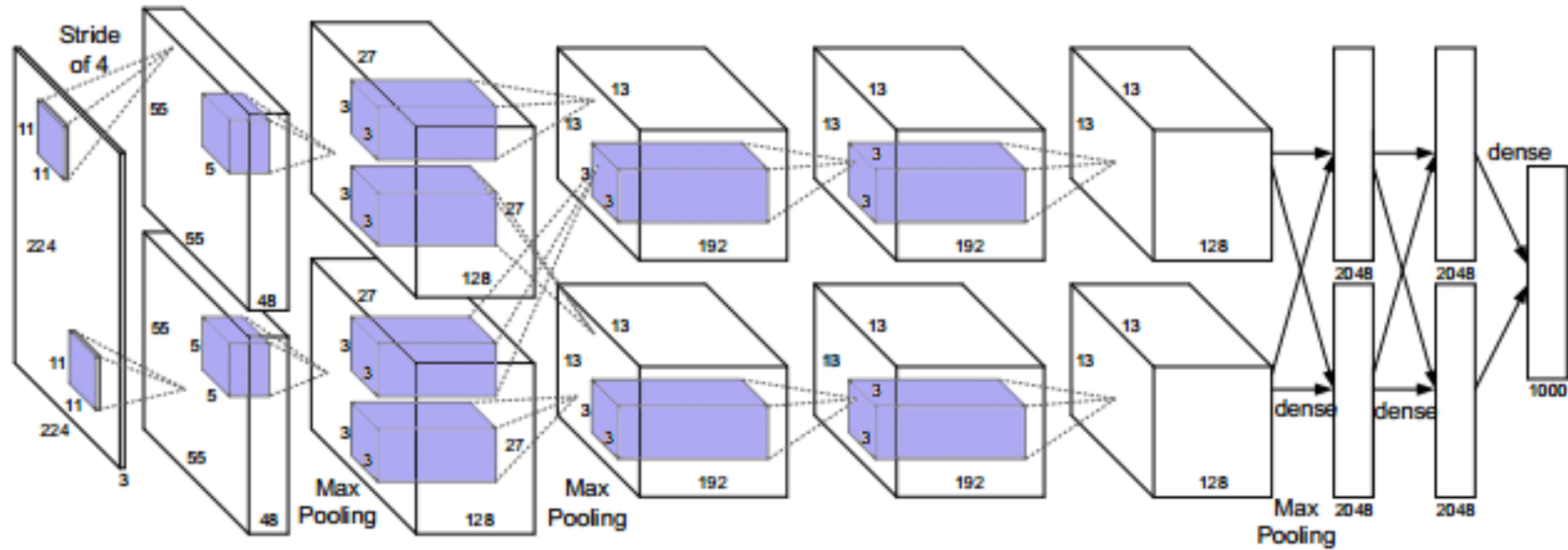
Filters



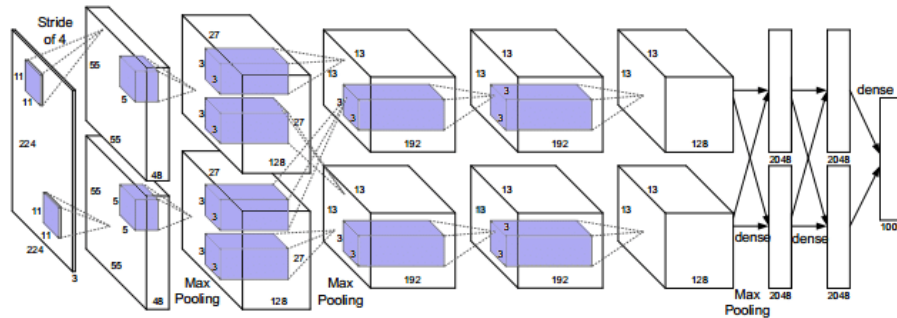
Filters



AlexNet (2012)

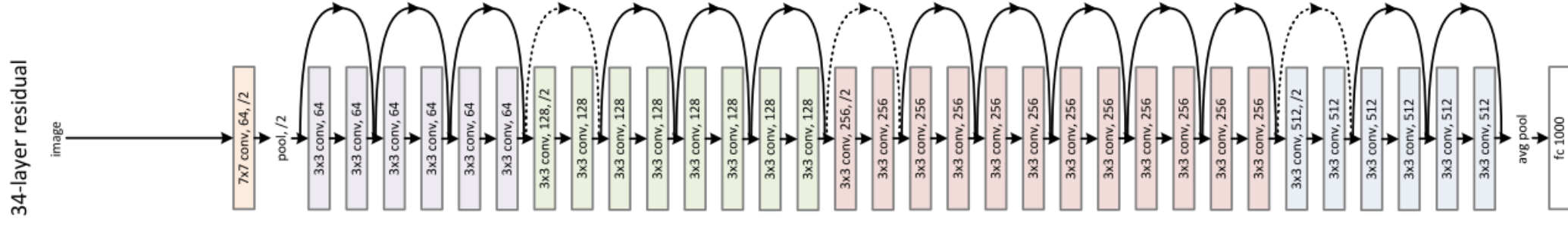


AlexNet (2012)

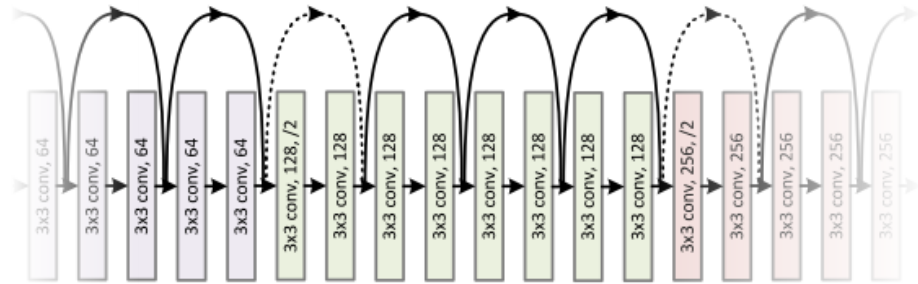


```
1 import torch.nn as nn
2
3 class AlexNet(nn.Module):
4     def __init__(self, num_classes: int = 1000, dropout: float = 0.5) -> None:
5         super().__init__()
6         _log_api_usage_once(self)
7         self.features = nn.Sequential(
8             nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
9             nn.ReLU(inplace=True),
10            nn.MaxPool2d(kernel_size=3, stride=2),
11            nn.Conv2d(64, 192, kernel_size=5, padding=2),
12            nn.ReLU(inplace=True),
13            nn.MaxPool2d(kernel_size=3, stride=2),
14            nn.Conv2d(192, 384, kernel_size=3, padding=1),
15            nn.ReLU(inplace=True),
16            nn.Conv2d(384, 256, kernel_size=3, padding=1),
17            nn.ReLU(inplace=True),
18            nn.Conv2d(256, 256, kernel_size=3, padding=1),
19            nn.ReLU(inplace=True),
20            nn.MaxPool2d(kernel_size=3, stride=2),
21        )
22        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
23        self.classifier = nn.Sequential(
24            nn.Dropout(p=dropout),
25            nn.Linear(256 * 6 * 6, 4096),
26            nn.ReLU(inplace=True),
27            nn.Dropout(p=dropout),
28            nn.Linear(4096, 4096),
29            nn.ReLU(inplace=True),
30            nn.Linear(4096, num_classes),
31        )
32
33    def forward(self, x: torch.Tensor) -> torch.Tensor:
34        x = self.features(x)
35        x = self.avgpool(x)
36        x = torch.flatten(x, 1)
37        x = self.classifier(x)
38        return x
```

ResNet (2015)

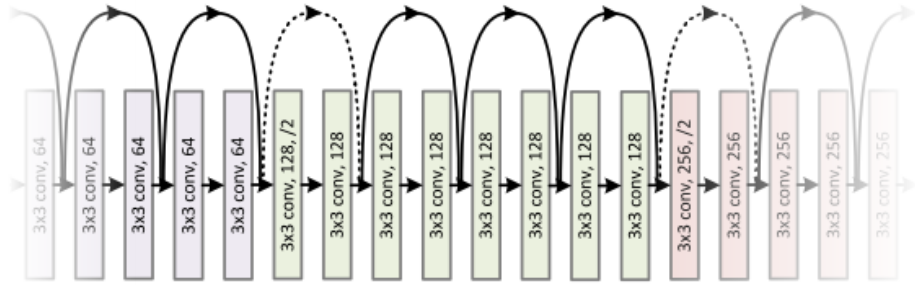


ResNet (2015)



```
1 class Bottleneck(nn.Module):
2     def __init__(self, *args, **kwargs) -> None:
3         self.conv1 = conv1x1(inplanes, width)
4         self.bn1 = norm_layer(width)
5         self.conv2 = conv3x3(width, width, stride, groups, dilation)
6         self.bn2 = norm_layer(width)
7         self.conv3 = conv1x1(width, planes * self.expansion)
8         self.bn3 = norm_layer(planes * self.expansion)
9         self.relu = nn.ReLU(inplace=True)
10        self.downsample = downsample
11        self.stride = stride
12
13    def forward(self, x: Tensor) -> Tensor:
14        identity = x
15
16        out = self.conv1(x)
17        out = self.bn1(out)
18        out = self.relu(out)
19
20        out = self.conv2(out)
21        out = self.bn2(out)
22        out = self.relu(out)
23
24        out = self.conv3(out)
25        out = self.bn3(out)
26
27        out += identity
28        out = self.relu(out)
29
30        return out
```

ResNet (2015)



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

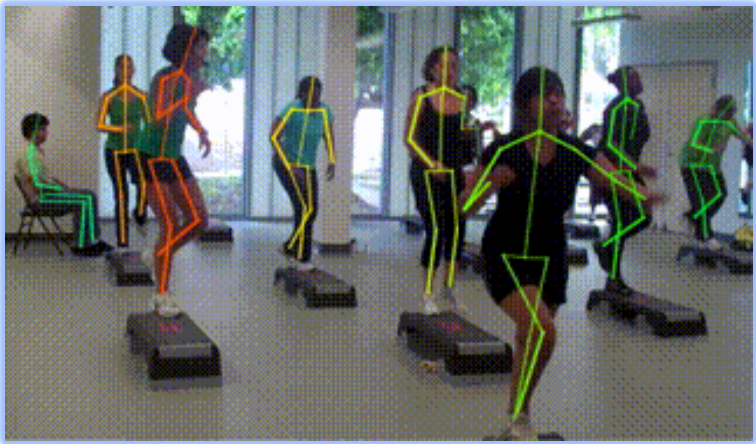
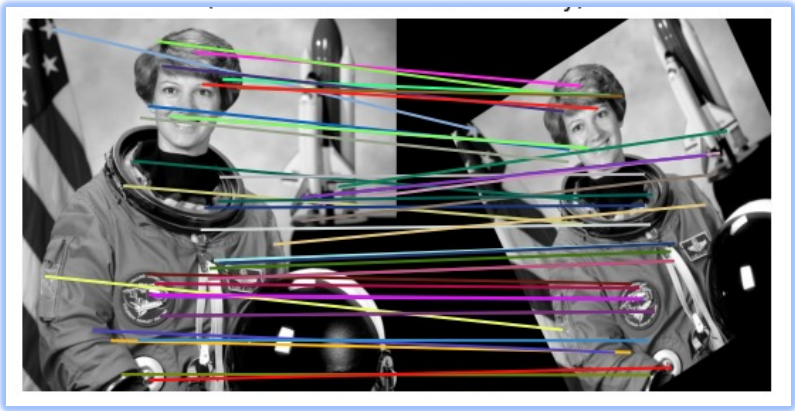
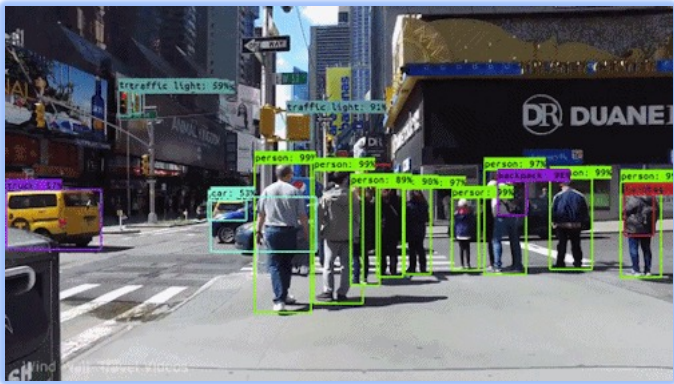
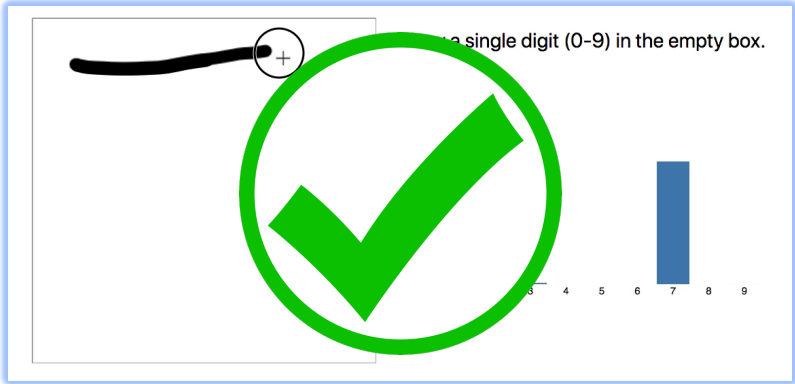
```

1 class Bottleneck(nn.Module):
2     def __init__(self, *args, **kwargs) -> None:
3         self.conv1 = conv1x1(inplanes, width)
4         self.bn1 = norm_layer(width)
5         self.conv2 = conv3x3(width, width, stride, groups, dilation)
6         self.bn2 = norm_layer(width)
7         self.conv3 = conv1x1(width, planes * self.expansion)
8         self.bn3 = norm_layer(planes * self.expansion)
9         self.relu = nn.ReLU(inplace=True)
10        self.downsample = downsample
11        self.stride = stride
12
13    def forward(self, x: Tensor) -> Tensor:
14        identity = x
15
16        out = self.conv1(x)
17        out = self.bn1(out)
18        out = self.relu(out)
19
20        out = self.conv2(out)
21        out = self.bn2(out)
22        out = self.relu(out)
23
24        out = self.conv3(out)
25        out = self.bn3(out)
26
27        out += identity
28        out = self.relu(out)
29
30    return out

```

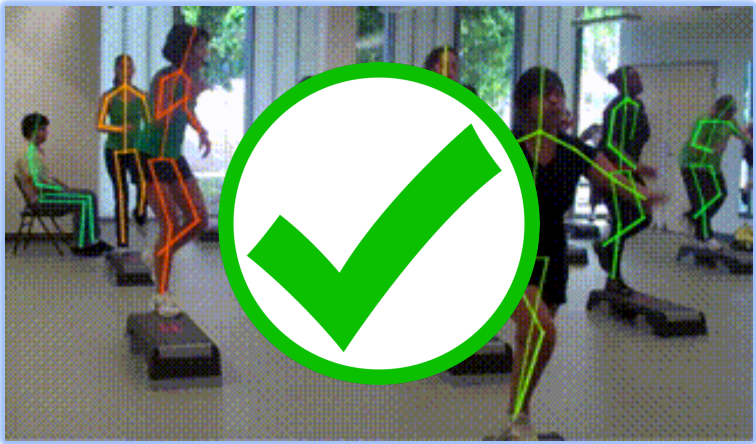
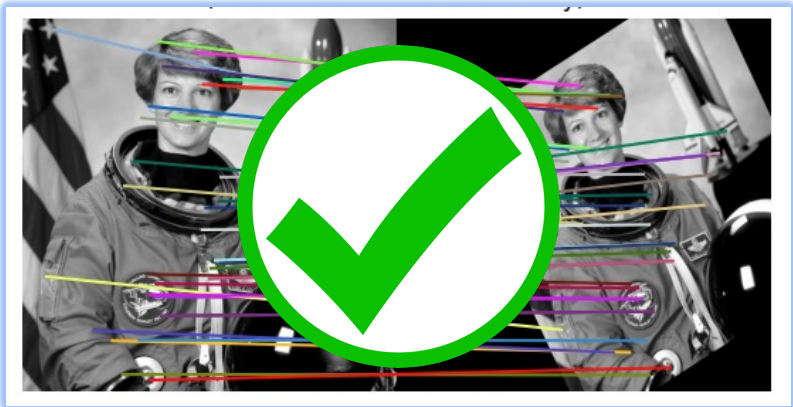
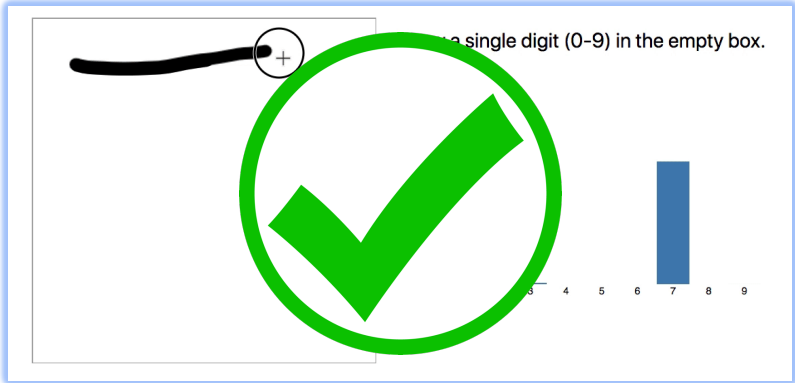
Computer Vision

A multitude of interesting challenges

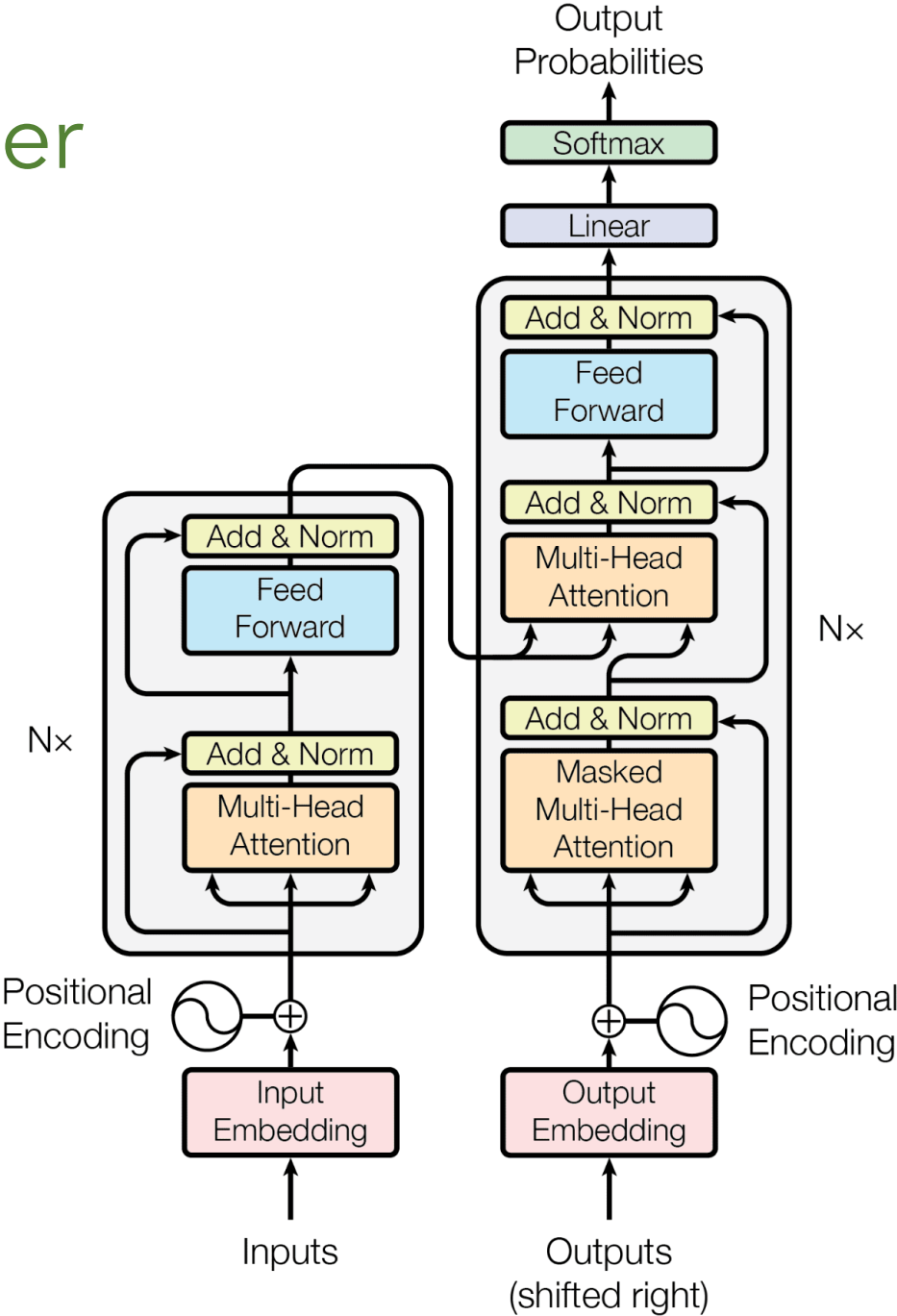


Computer Vision

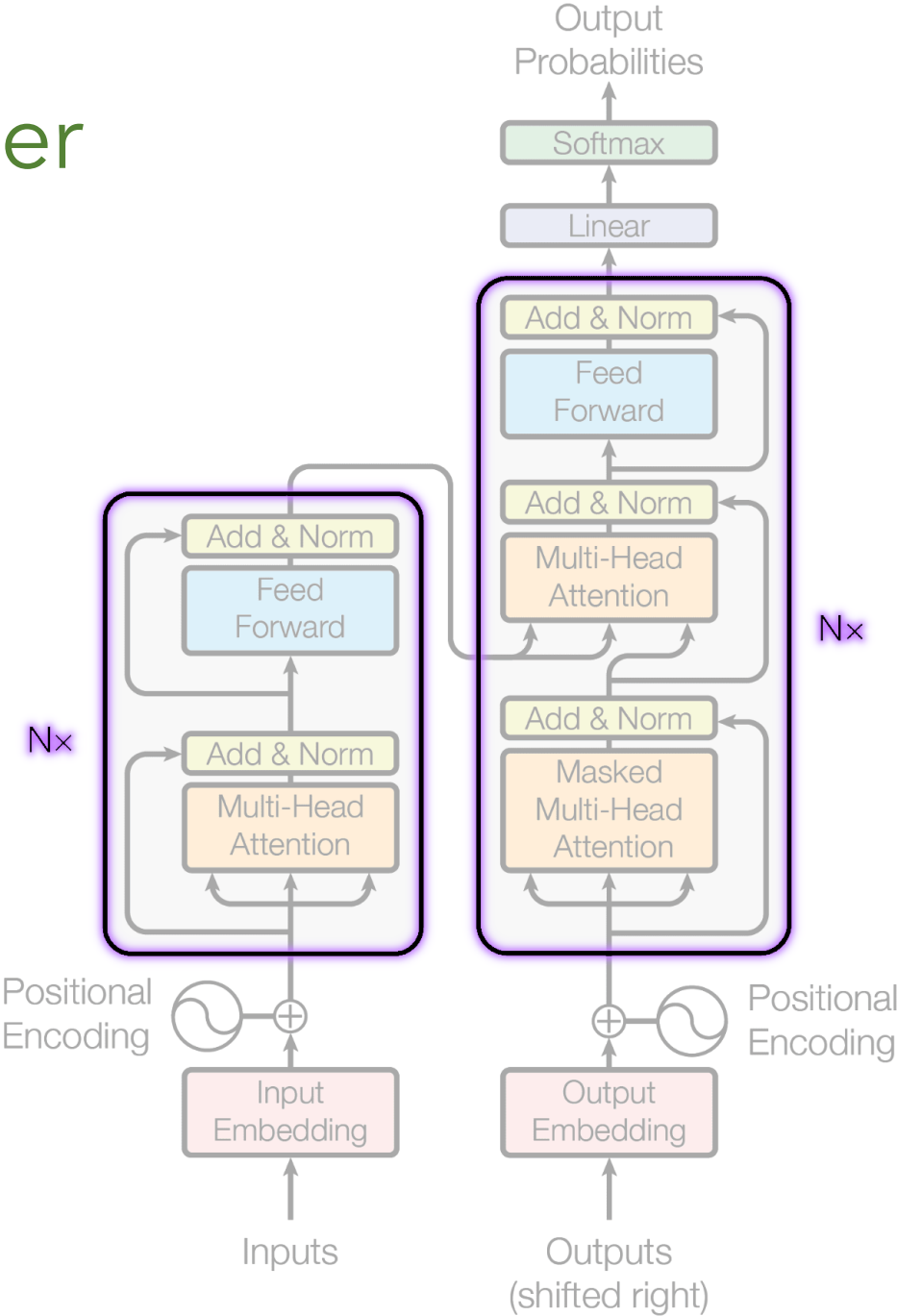
A multitude of interesting challenges



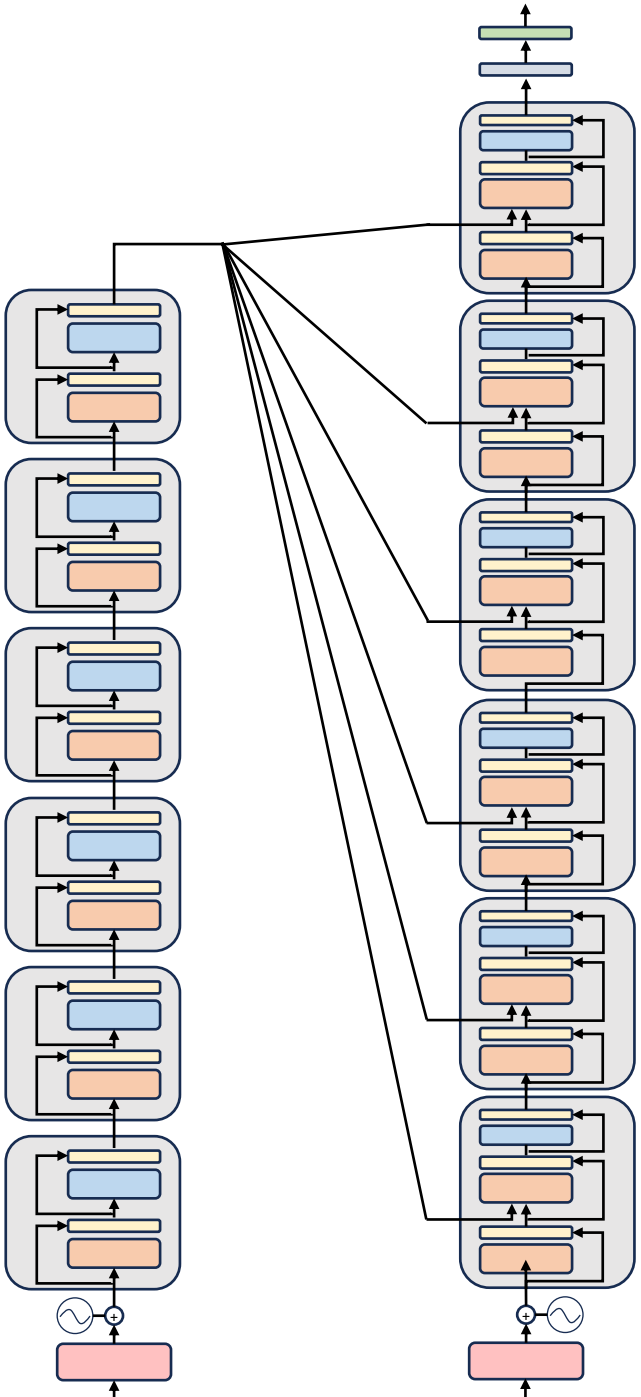
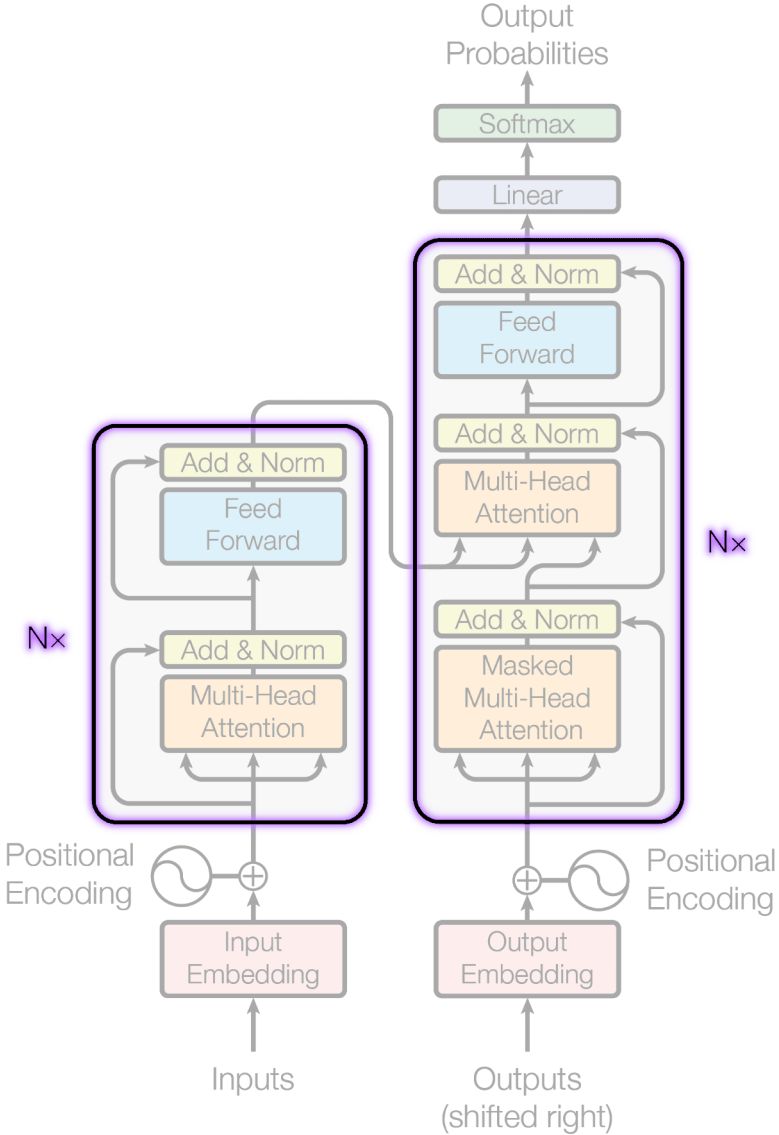
The Transformer



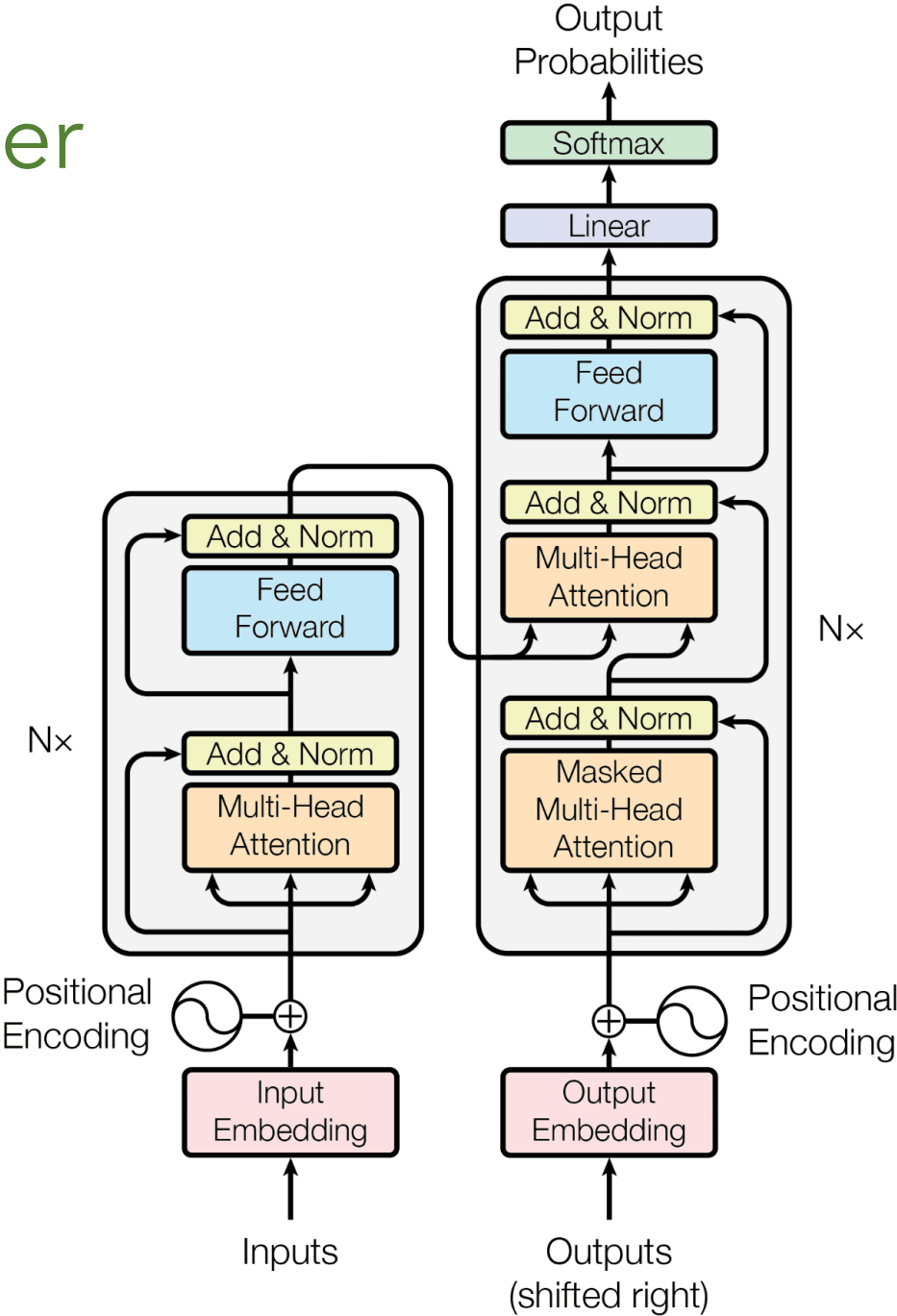
The Transformer



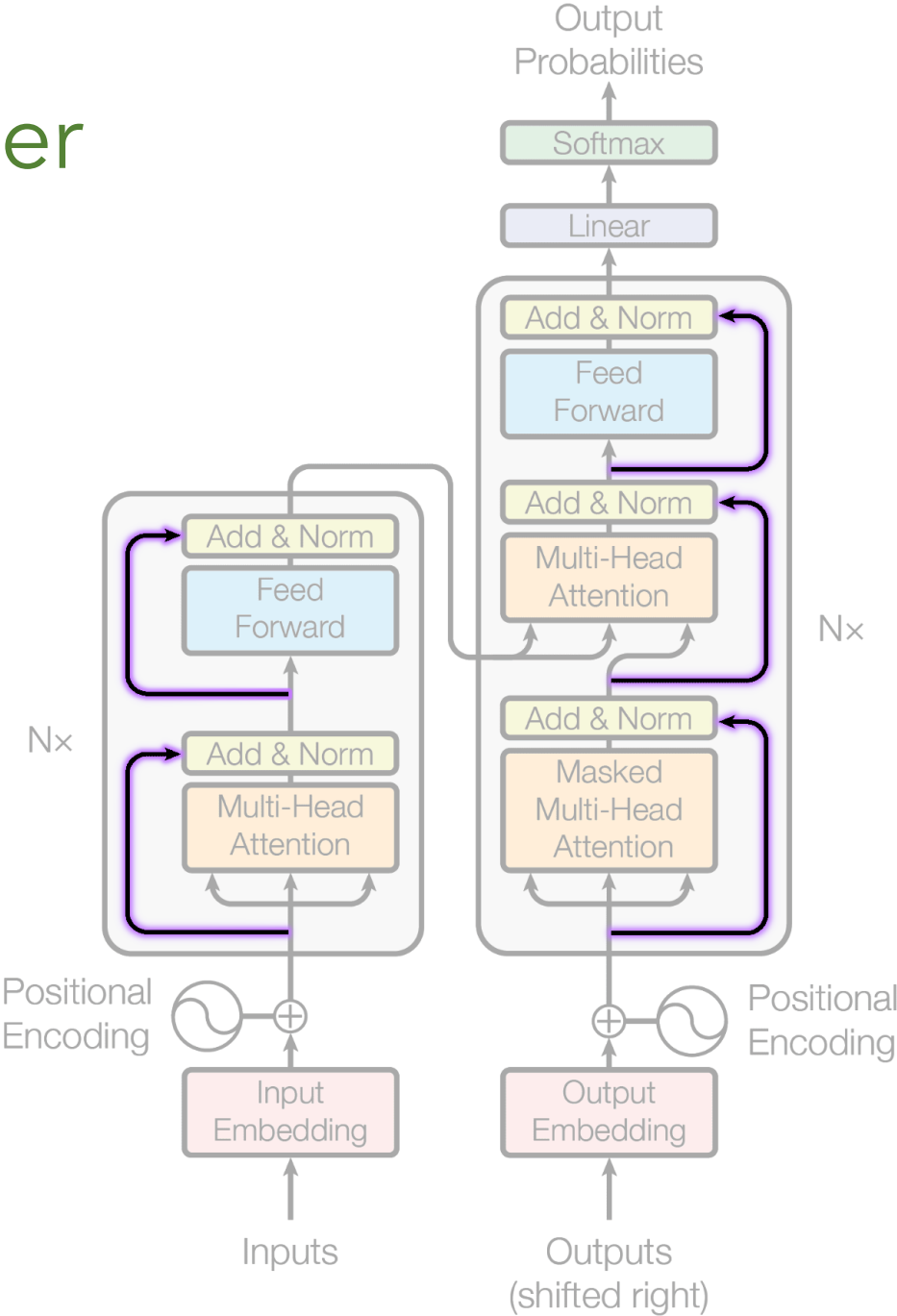
The Transformer



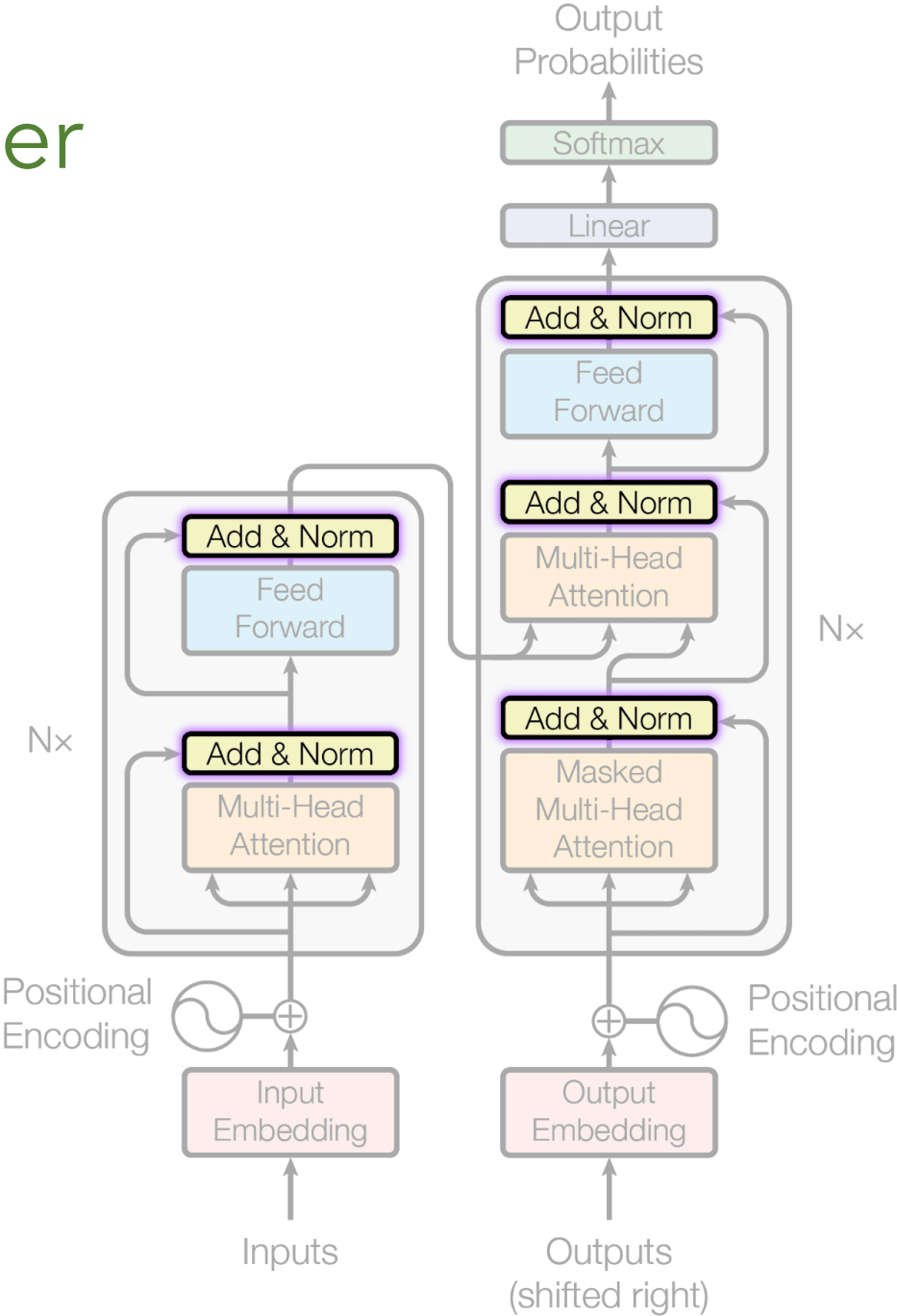
The Transformer



The Transformer



The Transformer



Natural Language Processing

Manipulating Human Language with Computers

Are there many interesting problems in this space?



Sequence to Sequence Modeling

E.g., Translation

Como un cordero al matadero



Like a lamb to the slaughter

Donde hay humo, hay fuego



Where there's smoke, there's fire

Nunca es tarde para aprender



It's never too late to learn

Que sera, sera



Whatever will be, will be

Representation: CV vs NLP



```
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 101
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 97
231 142 227 183 52 173 190 79 147 6 76 82 209 108 165 26 216 60 3 43 7 189
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76 3 230 159
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227 7 201 92
101 98 172 54 24 203 56 135 225 110 91 201 150 83 127 107 32 92 159 167
135 33 9 250 79 139 144 15 207 22 39 4 81 227 79 146 193 43 92 7 201
185 190 16 178 179 95 12 192 94 166 86 106 124 95 151 124 115 9 3 80 4 94
193 156 58 15 103 241 92 231 47 134 34 68 15 38 87 156 30 101 6 60 5 238
179 40 176 182 185 147 80 57 161 47 93 41 14 183 196 20 183 230 9 27 3 80
181 93 195 39 21 118 12 139 219 10 6 10 62 46 195 186 80 159 101 9 27
95 89 146 103 147 52 101 247 118 231 11 64 53 201 89 138 117 201 8 227 5 9
192 163 254 24 225 151 213 20 61 250 62 95 222 113 135 218 217 189 7 189 3 230
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 5 9 6 60
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 97 3 43
92 239 19 112 109 150 53 114 24 235 135 68 255 222 216 109 224 94 97 3 43
177 155 120 161 81 152 36 54 3 182 35 109 9 89 21 210 169 27 76 2 105
211 243 121 147 66 71 118 140 210 139 116 21 16 129 59 95 26 238 4 94
233 195 239 23 56 140 122 174 72 169 30 175 53 166 131 182 78 97 238
193 133 247 156 130 88 217 197 13 203 53 246 109 43 145 119 193 80
```

Representation: CV vs NLP



```
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 101
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 97
231 142 227 183 52 173 190 79 147 6 76 82 209 108 165 26 216 60 3 43 7 189
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76 3 230 159
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227 7 201 92
101 98 172 54 24 203 56 135 225 110 91 201 150 83 127 107 32 92 159 167
135 33 9 250 79 139 144 15 207 22 39 4 81 227 79 146 193 43 92 7 201
185 190 16 178 179 95 12 192 94 166 86 106 124 95 151 124 115 9 3 80 4 94
193 156 58 15 103 241 92 231 47 134 34 68 15 38 87 156 30 101 6 60 5 238
179 40 176 182 185 147 80 57 161 47 93 41 14 183 196 20 183 230 9 27 3 80
181 93 195 39 21 118 12 139 219 10 6 10 62 46 195 186 80 159 101 9 27
95 89 146 103 147 52 101 247 118 231 11 64 53 201 89 138 117 201 8 227 5 9
192 163 254 24 225 151 213 20 61 250 62 95 222 113 135 218 217 189 7 189 3 230
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 5 9 6 60
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 9 6 60
92 239 19 112 109 150 53 114 24 235 135 68 255 222 216 109 224 94 3 97 3 43
177 155 120 161 81 152 36 54 3 182 35 109 9 89 21 210 169 27 76 2 105
211 243 121 147 66 71 118 140 210 139 116 21 16 129 59 95 26 238 4 94
233 195 239 23 56 140 122 174 72 169 30 175 53 166 131 182 78 97 5 238
193 133 247 156 130 88 217 197 13 203 53 246 109 43 145 119 193 80
```

```
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 101
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 97
231 142 227 183 52 173 190 79 147 6 76 82 209 108 165 26 216 60 3 43 7 189
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76 3 230 159
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227 7 201 92
101 98 172 54 24 203 56 135 225 110 91 201 150 83 127 107 32 92 159 167
135 33 9 250 79 139 144 15 207 22 39 4 81 227 79 146 193 43 92 7 201
185 190 16 178 179 95 12 192 94 166 86 106 124 95 151 124 115 9 3 80 4 94
193 156 58 15 103 241 92 231 47 134 34 68 15 38 87 156 30 101 6 60 5 238
179 40 176 182 185 147 80 57 161 47 93 41 14 183 196 20 183 230 9 27 3 80
181 93 195 39 21 118 12 139 219 10 6 10 62 46 195 186 80 159 101 9 27
95 89 146 103 147 52 101 247 118 231 11 64 53 201 89 138 117 201 8 227 5 9
192 163 254 24 225 151 213 20 61 250 62 95 222 113 135 218 217 189 7 189 3 230
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 5 9 6 60
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 9 6 60
92 239 19 112 109 150 53 114 24 235 135 68 255 222 216 109 224 94 3 97 3 43
177 155 120 161 81 152 36 54 3 182 35 109 9 89 21 210 169 27 76 2 105
211 243 121 147 66 71 118 140 210 139 116 21 16 129 59 95 26 238 4 94
233 195 239 23 56 140 122 174 72 169 30 175 53 166 131 182 78 97 5 238
193 133 247 156 130 88 217 197 13 203 53 246 109 43 145 119 193 80
```

Representation: CV vs NLP



slaughter

73 6C 61 75 67 68 74 65 72

9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 101
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 97
231 142 227 183 52 173 190 79 147 6 76 82 209 108 165 26 216 60 3 43 7 189
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76 3 230 159
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227 7 201 92
101 98 172 54 24 203 56 135 225 110 91 201 150 83 127 107 32 92 159 167
135 33 9 250 79 139 144 15 207 22 39 4 81 227 79 146 193 43 92 7 201
185 190 16 178 179 95 12 192 94 166 86 106 124 95 151 124 115 9 3 80 4 94
193 156 58 15 103 241 92 231 47 134 34 68 15 38 87 156 30 101 6 60 5 238
179 40 176 182 185 147 80 57 161 47 93 41 14 183 196 20 183 230 9 27 3 80
181 93 195 39 21 118 12 139 219 10 6 10 62 46 195 186 80 159 101 9 27
95 89 146 103 147 52 101 247 118 231 11 64 53 201 89 138 117 201 8 227 5 9
192 163 254 24 225 151 213 20 61 250 62 95 222 113 135 218 217 189 7 189 3 230
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 5 9 6 60
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 9 6 60
92 239 19 112 109 150 53 114 24 235 135 68 255 222 216 109 224 94 3 97 3 43
177 155 120 161 81 152 36 54 3 182 35 109 9 89 21 210 169 27 76 2 105
211 243 121 147 66 71 118 140 210 139 116 21 16 129 59 95 26 238 4 94
233 195 239 23 56 140 122 174 72 169 30 175 53 166 131 182 78 97 5 238
193 133 247 156 130 88 217 197 13 203 53 246 109 43 145 119 193 80

9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 101
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 97
231 142 227 183 52 173 190 79 147 6 76 82 209 108 165 26 216 60 3 43 7 189
149 215 139 9 145 10 38 160 59 83 242 230 93 206 193 110 2 76 3 230 159
9 158 113 8 170 231 251 235 131 220 242 73 128 198 216 86 188 227 7 201 92
101 98 172 54 24 203 56 135 225 110 91 201 150 83 127 107 32 92 159 167
135 33 9 250 79 139 144 15 207 22 39 4 81 227 79 146 193 43 92 7 201
185 190 16 178 179 95 12 192 94 166 86 106 124 95 151 124 115 9 3 80 4 94
193 156 58 15 103 241 92 231 47 134 34 68 15 38 87 156 30 101 6 60 5 238
179 40 176 182 185 147 80 57 161 47 93 41 14 183 196 20 183 230 9 27 3 80
181 93 195 39 21 118 12 139 219 10 6 10 62 46 195 186 80 159 101 9 27
95 89 146 103 147 52 101 247 118 231 11 64 53 201 89 138 117 201 8 227 5 9
192 163 254 24 225 151 213 20 61 250 62 95 222 113 135 218 217 189 7 189 3 230
73 6 35 59 197 191 154 210 227 10 213 240 75 90 104 24 99 167 5 9 6 60
150 30 149 253 48 220 212 172 251 20 200 180 21 192 61 98 172 105 9 6 60
92 239 19 112 109 150 53 114 24 235 135 68 255 222 216 109 224 94 3 97 3 43
177 155 120 161 81 152 36 54 3 182 35 109 9 89 21 210 169 27 76 2 105
211 243 121 147 66 71 118 140 210 139 116 21 16 129 59 95 26 238 4 94
233 195 239 23 56 140 122 174 72 169 30 175 53 166 131 182 78 97 5 238
193 133 247 156 130 88 217 197 13 203 53 246 109 43 145 119 193 80

slaughter

73 6C 61 75 67 68 74 65 72

Embedding

Encoding Words into Vectors

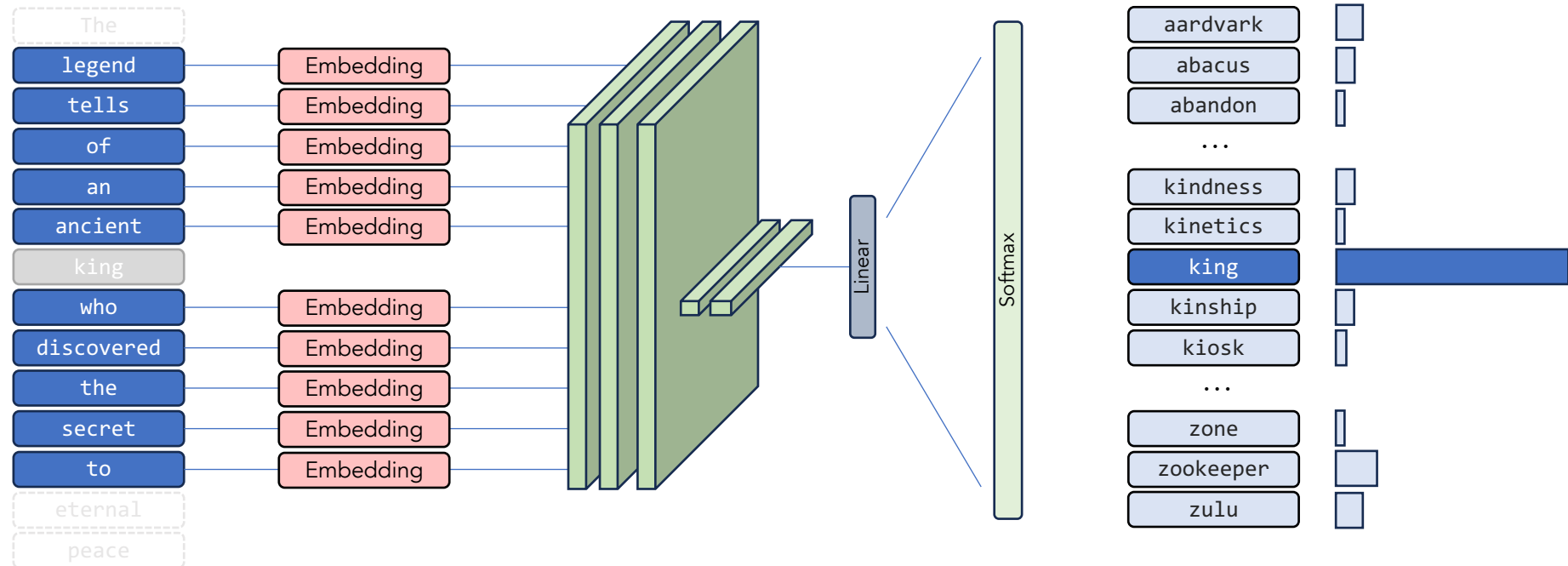
king \rightarrow `np.array([0.603 0.198 -0.585 ... -0.074])`

Embedding

Encoding Words into Vectors

king \rightarrow `np.array([0.603 0.198 -0.585 ... -0.074])`

“The legend tells of an ancient king who discovered the secret to eternal peace.”



Embedding

Encoding Words into Vectors

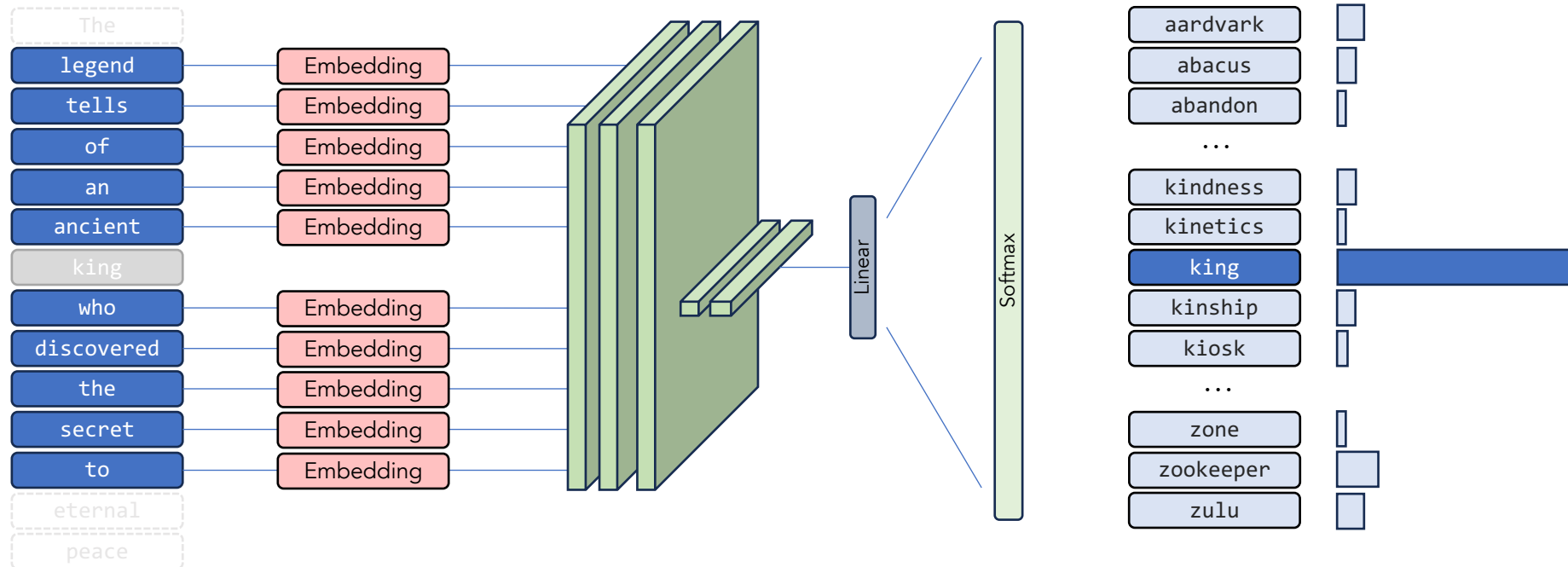
king \rightarrow `np.array([0.603 0.198 -0.585 ... -0.074])`

queen \rightarrow `np.array([1.298 1.422 -1.122 ... 0.244])`

man \rightarrow `np.array([0.252 -0.836 -0.294 ... 0.455])`

woman \rightarrow `np.array([0.947 0.388 -0.831 ... 0.773])`

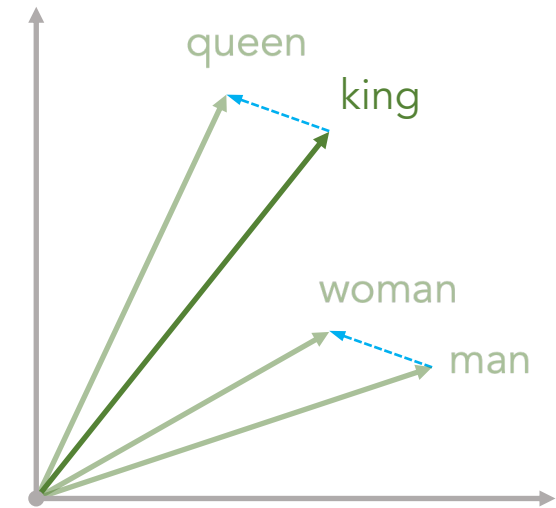
“The legend tells of an ancient king who discovered the secret to eternal peace.”



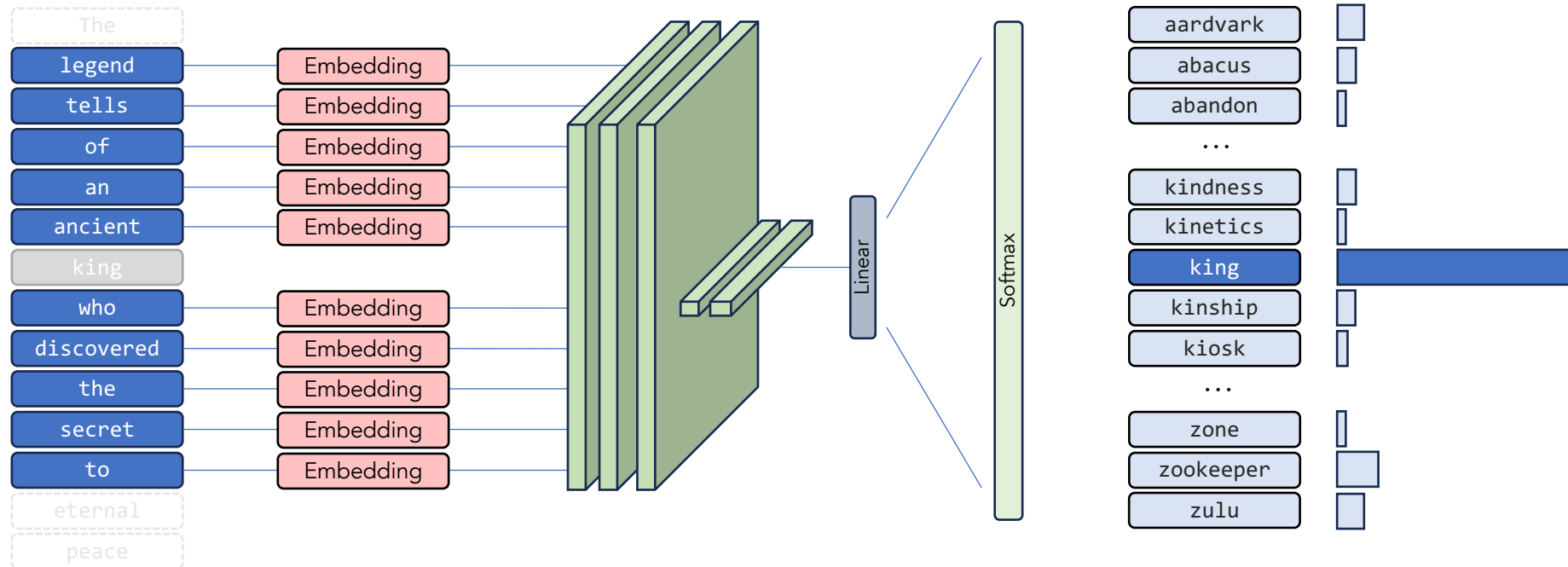
Embedding

Encoding Words into Vectors

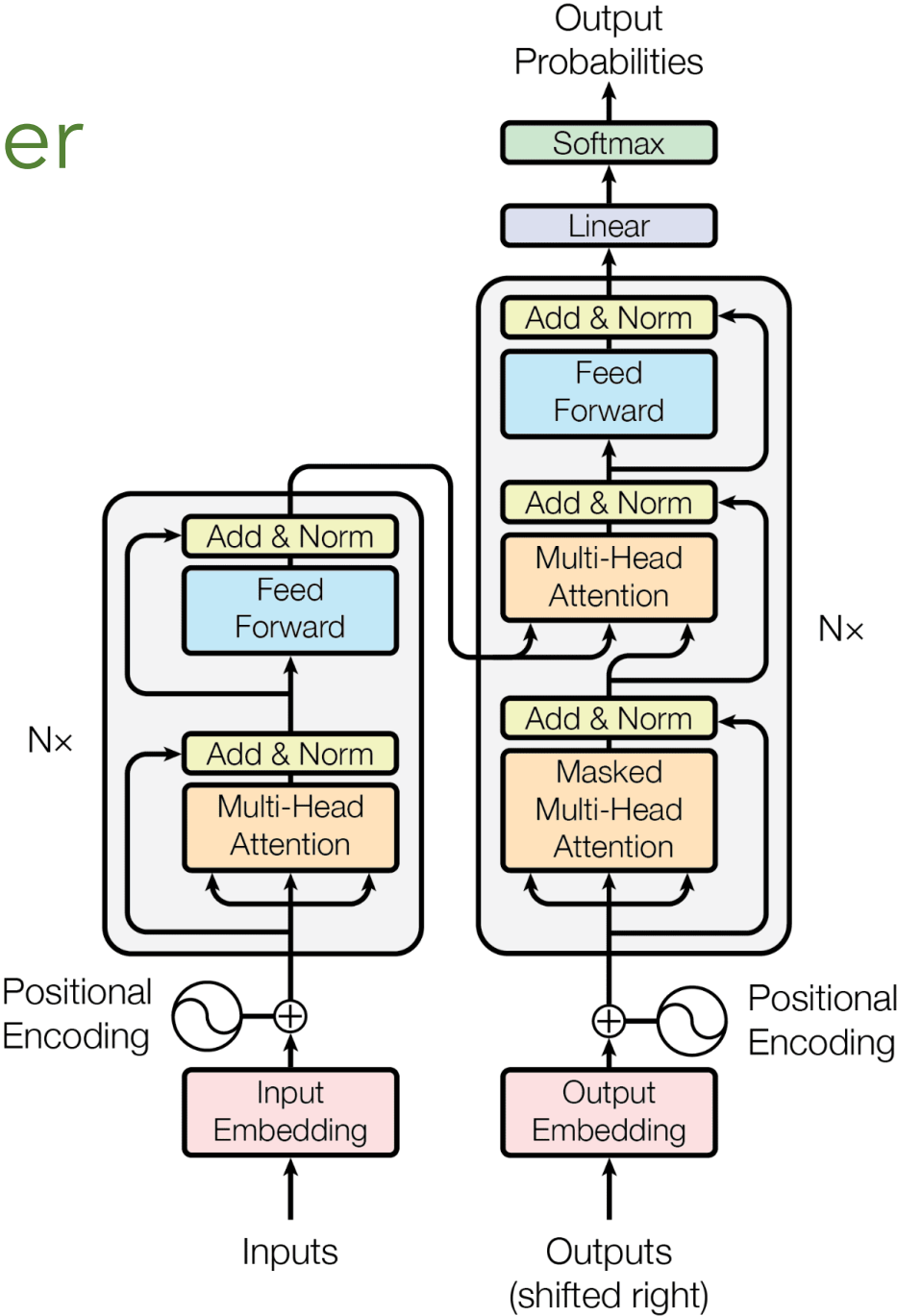
king \rightarrow `np.array([0.603 0.198 -0.585 ... -0.074])`
queen \rightarrow `np.array([1.298 1.422 -1.122 ... 0.244])`
man \rightarrow `np.array([0.252 -0.836 -0.294 ... 0.455])`
woman \rightarrow `np.array([0.947 0.388 -0.831 ... 0.773])`



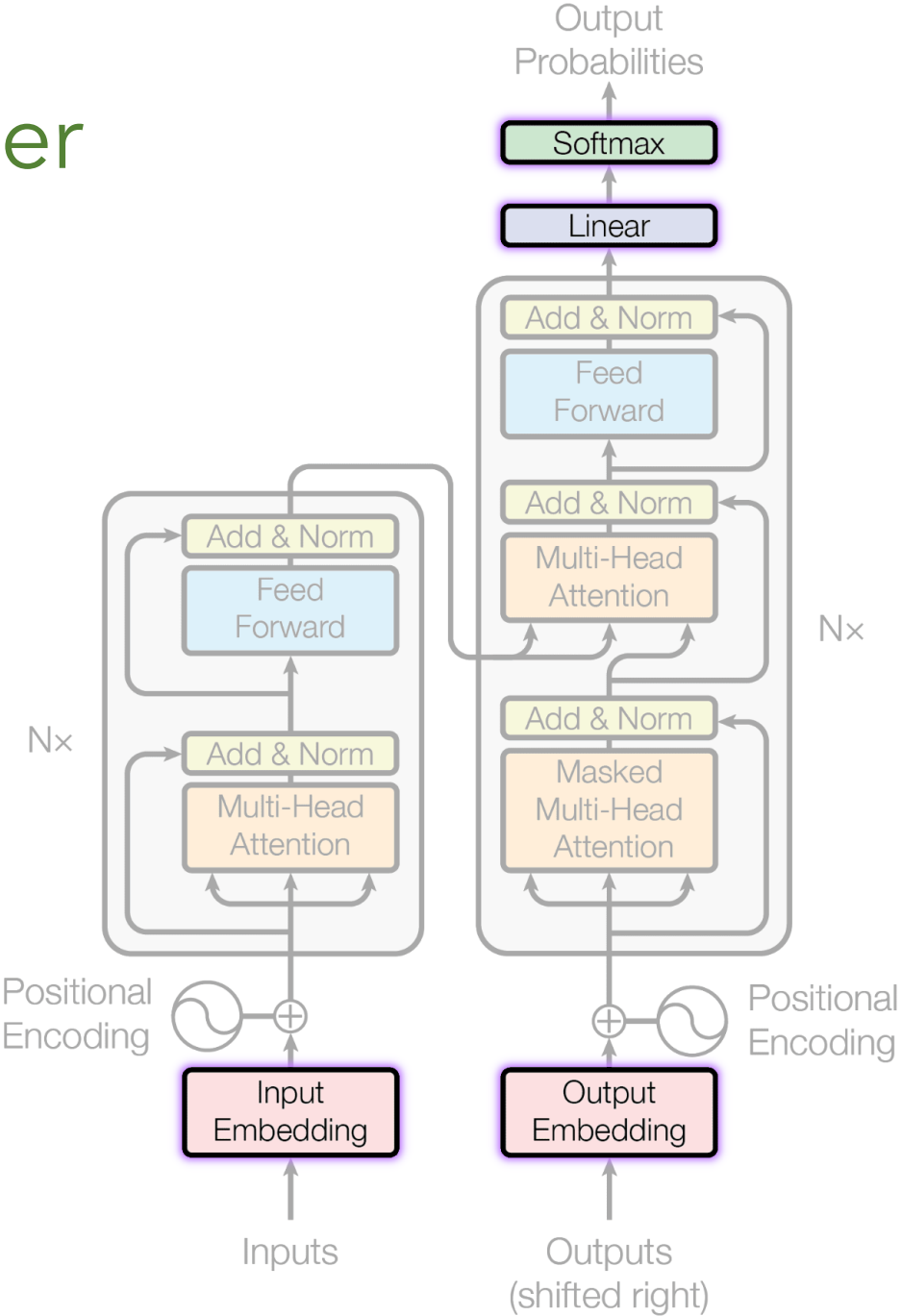
“The legend tells of an ancient king who discovered the secret to eternal peace.”



The Transformer



The Transformer



Sequence to Sequence Modeling

E.g., Translation

Como un cordero al matadero



Like a lamb to the slaughter

Donde hay humo, hay fuego



Where there's smoke, there's fire

Nunca es tarde para aprender



It's never too late to learn

Que sera, sera



Whatever will be, will be

Sequence to Sequence Modeling

E.g., Translation

Como un cordero al matadero

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.189 \\ -0.602 \\ -0.882 \end{bmatrix} \begin{bmatrix} -0.111 \\ -0.711 \\ 0.422 \end{bmatrix} \begin{bmatrix} 0.308 \\ 0.647 \\ 0.305 \end{bmatrix} \begin{bmatrix} 0.491 \\ 0.480 \\ 0.502 \end{bmatrix}$$


Like a lamb to the slaughter

Donde hay humo, hay fuego

$$\begin{bmatrix} -0.508 \\ 0.339 \\ -0.690 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.937 \\ -0.322 \\ 0.916 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.178 \\ -0.945 \\ -0.829 \end{bmatrix}$$


Where there's smoke, there's fire

Nunca es tarde para aprender

$$\begin{bmatrix} 0.921 \\ 0.412 \\ 0.376 \end{bmatrix} \begin{bmatrix} -0.928 \\ -0.751 \\ 0.811 \end{bmatrix} \begin{bmatrix} 0.847 \\ -0.445 \\ -0.988 \end{bmatrix} \begin{bmatrix} -0.952 \\ 0.484 \\ -0.867 \end{bmatrix} \begin{bmatrix} -0.365 \\ 0.498 \\ 0.310 \end{bmatrix}$$


It's never too late to learn

Que sera, sera

$$\begin{bmatrix} 0.989 \\ 0.993 \\ 0.044 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix}$$


Whatever will be, will be

Sequence to Sequence Modeling

E.g., Translation

Como un cordero al matadero

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.189 \\ -0.602 \\ -0.882 \end{bmatrix} \begin{bmatrix} -0.111 \\ -0.711 \\ 0.422 \end{bmatrix} \begin{bmatrix} 0.308 \\ 0.647 \\ 0.305 \end{bmatrix} \begin{bmatrix} 0.491 \\ 0.480 \\ 0.502 \end{bmatrix}$$


Like a lamb to the slaughter

Donde hay humo, hay fuego

$$\begin{bmatrix} -0.508 \\ 0.339 \\ -0.690 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.937 \\ -0.322 \\ 0.916 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.178 \\ -0.945 \\ -0.829 \end{bmatrix}$$


Where there's smoke, there's fire

Nunca es tarde para aprender

$$\begin{bmatrix} 0.921 \\ 0.412 \\ 0.376 \end{bmatrix} \begin{bmatrix} -0.928 \\ -0.751 \\ 0.811 \end{bmatrix} \begin{bmatrix} 0.847 \\ -0.445 \\ -0.988 \end{bmatrix} \begin{bmatrix} -0.952 \\ 0.484 \\ -0.867 \end{bmatrix} \begin{bmatrix} -0.365 \\ 0.498 \\ 0.310 \end{bmatrix}$$


It's never too late to learn

Que sera, sera <PAD> <PAD>

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$


Whatever will be, will be

Sequence to Sequence Modeling

E.g., Translation

Como un cordero al matadero

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.189 \\ -0.602 \\ -0.882 \end{bmatrix} \begin{bmatrix} -0.111 \\ -0.711 \\ 0.422 \end{bmatrix} \begin{bmatrix} 0.308 \\ 0.647 \\ 0.305 \end{bmatrix} \begin{bmatrix} 0.491 \\ 0.480 \\ 0.502 \end{bmatrix}$$


Like a lamb to the slaughter

$$\begin{bmatrix} -0.820 \\ -0.553 \\ -0.315 \end{bmatrix} \begin{bmatrix} 0.393 \\ -0.208 \\ 0.690 \end{bmatrix} \begin{bmatrix} 0.104 \\ -0.116 \\ 0.204 \end{bmatrix} \begin{bmatrix} -0.788 \\ 0.497 \\ 0.909 \end{bmatrix} \begin{bmatrix} 0.608 \\ -0.783 \\ 0.117 \end{bmatrix} \begin{bmatrix} -0.100 \\ 0.683 \\ 0.603 \end{bmatrix}$$

Donde hay humo, hay fuego

$$\begin{bmatrix} -0.508 \\ 0.339 \\ -0.690 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.937 \\ -0.322 \\ 0.916 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.178 \\ -0.945 \\ -0.829 \end{bmatrix}$$


Where there's smoke, there's fire

$$\begin{bmatrix} -0.359 \\ -0.023 \\ 0.355 \end{bmatrix} \begin{bmatrix} -0.066 \\ -0.431 \\ 0.829 \end{bmatrix} \begin{bmatrix} 0.662 \\ 0.419 \\ 0.871 \end{bmatrix} \begin{bmatrix} -0.066 \\ -0.431 \\ 0.829 \end{bmatrix} \begin{bmatrix} -0.128 \\ -0.461 \\ -0.123 \end{bmatrix}$$

Nunca es tarde para aprender

$$\begin{bmatrix} 0.921 \\ 0.412 \\ 0.376 \end{bmatrix} \begin{bmatrix} -0.928 \\ -0.751 \\ 0.811 \end{bmatrix} \begin{bmatrix} 0.847 \\ -0.445 \\ -0.988 \end{bmatrix} \begin{bmatrix} -0.952 \\ 0.484 \\ -0.867 \end{bmatrix} \begin{bmatrix} -0.365 \\ 0.498 \\ 0.310 \end{bmatrix}$$


It's never too late to learn

$$\begin{bmatrix} -0.194 \\ -0.118 \\ -0.049 \end{bmatrix} \begin{bmatrix} -0.478 \\ -0.072 \\ -0.431 \end{bmatrix} \begin{bmatrix} -0.775 \\ -0.156 \\ 0.371 \end{bmatrix} \begin{bmatrix} -0.066 \\ -0.431 \\ 0.829 \end{bmatrix} \begin{bmatrix} -0.788 \\ 0.497 \\ 0.909 \end{bmatrix} \begin{bmatrix} -0.327 \\ 0.614 \\ -0.816 \end{bmatrix}$$

Que sera, sera <PAD> <PAD>

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$


Whatever will be, will be

$$\begin{bmatrix} -0.405 \\ -0.351 \\ 0.483 \end{bmatrix} \begin{bmatrix} -0.138 \\ 0.769 \\ 0.895 \end{bmatrix} \begin{bmatrix} -0.830 \\ -0.269 \\ -0.737 \end{bmatrix} \begin{bmatrix} -0.138 \\ 0.769 \\ 0.895 \end{bmatrix} \begin{bmatrix} -0.830 \\ -0.269 \\ -0.737 \end{bmatrix}$$

Sequence to Sequence Modeling

E.g., Translation

Como un cordero al matadero

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.189 \\ -0.602 \\ -0.882 \end{bmatrix} \begin{bmatrix} -0.111 \\ -0.711 \\ 0.422 \end{bmatrix} \begin{bmatrix} 0.308 \\ 0.647 \\ 0.305 \end{bmatrix} \begin{bmatrix} 0.491 \\ 0.480 \\ 0.502 \end{bmatrix}$$


Like a lamb to the slaughter

$$\begin{bmatrix} -0.820 \\ -0.553 \\ -0.315 \end{bmatrix} \begin{bmatrix} 0.393 \\ -0.208 \\ 0.690 \end{bmatrix} \begin{bmatrix} 0.104 \\ -0.116 \\ 0.204 \end{bmatrix} \begin{bmatrix} -0.788 \\ 0.497 \\ 0.909 \end{bmatrix} \begin{bmatrix} 0.608 \\ -0.783 \\ 0.117 \end{bmatrix} \begin{bmatrix} -0.100 \\ 0.683 \\ 0.603 \end{bmatrix}$$

Donde hay humo, hay fuego

$$\begin{bmatrix} -0.508 \\ 0.339 \\ -0.690 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.937 \\ -0.322 \\ 0.916 \end{bmatrix} \begin{bmatrix} 0.180 \\ -0.693 \\ -0.907 \end{bmatrix} \begin{bmatrix} -0.178 \\ -0.945 \\ -0.829 \end{bmatrix}$$


Where there's smoke, there's fire <PAD>

$$\begin{bmatrix} -0.359 \\ -0.023 \\ 0.355 \end{bmatrix} \begin{bmatrix} -0.066 \\ -0.431 \\ 0.829 \end{bmatrix} \begin{bmatrix} 0.662 \\ 0.419 \\ 0.871 \end{bmatrix} \begin{bmatrix} -0.066 \\ -0.431 \\ 0.829 \end{bmatrix} \begin{bmatrix} -0.128 \\ -0.461 \\ -0.123 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$

Nunca es tarde para aprender

$$\begin{bmatrix} 0.921 \\ 0.412 \\ 0.376 \end{bmatrix} \begin{bmatrix} -0.928 \\ -0.751 \\ 0.811 \end{bmatrix} \begin{bmatrix} 0.847 \\ -0.445 \\ -0.988 \end{bmatrix} \begin{bmatrix} -0.952 \\ 0.484 \\ -0.867 \end{bmatrix} \begin{bmatrix} -0.365 \\ 0.498 \\ 0.310 \end{bmatrix}$$


It's never too late to learn

$$\begin{bmatrix} -0.194 \\ -0.118 \\ -0.049 \end{bmatrix} \begin{bmatrix} -0.478 \\ -0.072 \\ -0.431 \end{bmatrix} \begin{bmatrix} -0.775 \\ -0.156 \\ 0.371 \end{bmatrix} \begin{bmatrix} -0.066 \\ -0.431 \\ 0.829 \end{bmatrix} \begin{bmatrix} -0.788 \\ 0.497 \\ 0.909 \end{bmatrix} \begin{bmatrix} -0.327 \\ 0.614 \\ -0.816 \end{bmatrix}$$

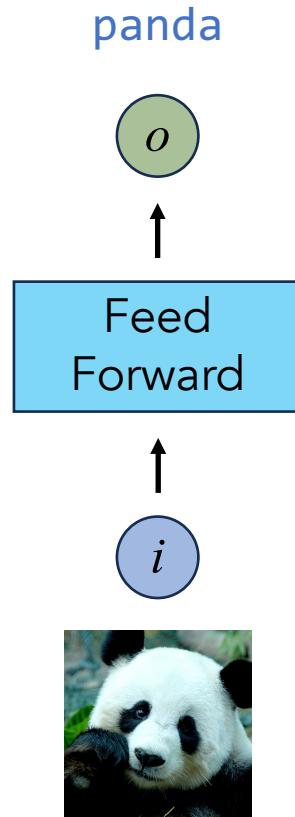
Que sera, sera <PAD> <PAD>

$$\begin{bmatrix} -0.698 \\ 0.729 \\ 0.362 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.446 \\ -0.529 \\ 0.714 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$

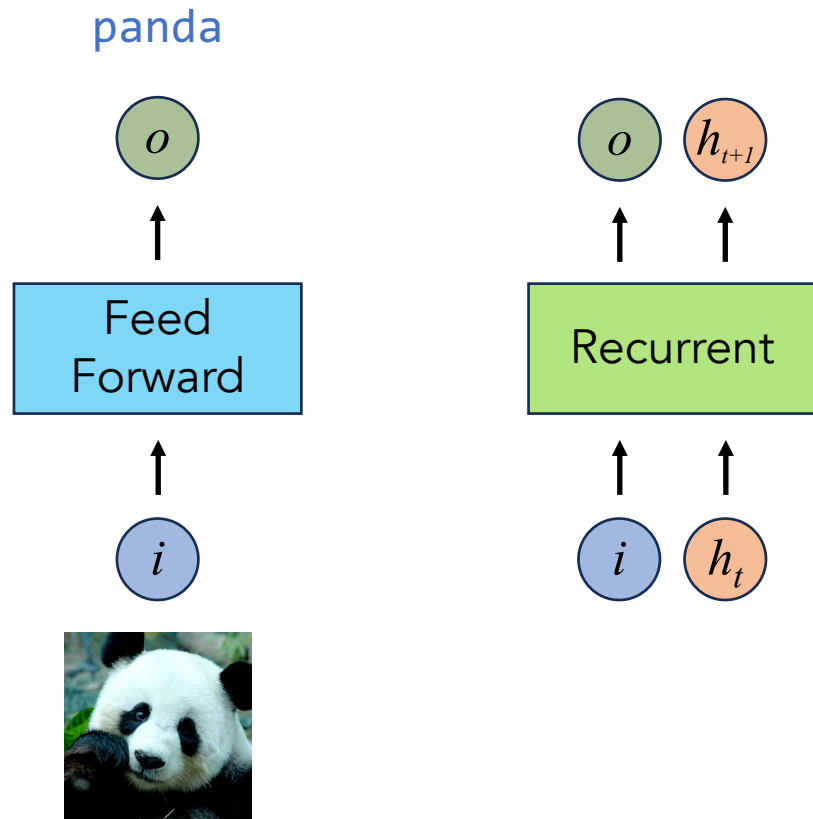

Whatever will be, will be <PAD>

$$\begin{bmatrix} -0.405 \\ -0.351 \\ 0.483 \end{bmatrix} \begin{bmatrix} -0.138 \\ 0.769 \\ 0.895 \end{bmatrix} \begin{bmatrix} -0.830 \\ -0.269 \\ -0.737 \end{bmatrix} \begin{bmatrix} -0.138 \\ 0.769 \\ 0.895 \end{bmatrix} \begin{bmatrix} -0.830 \\ -0.269 \\ -0.737 \end{bmatrix} \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$

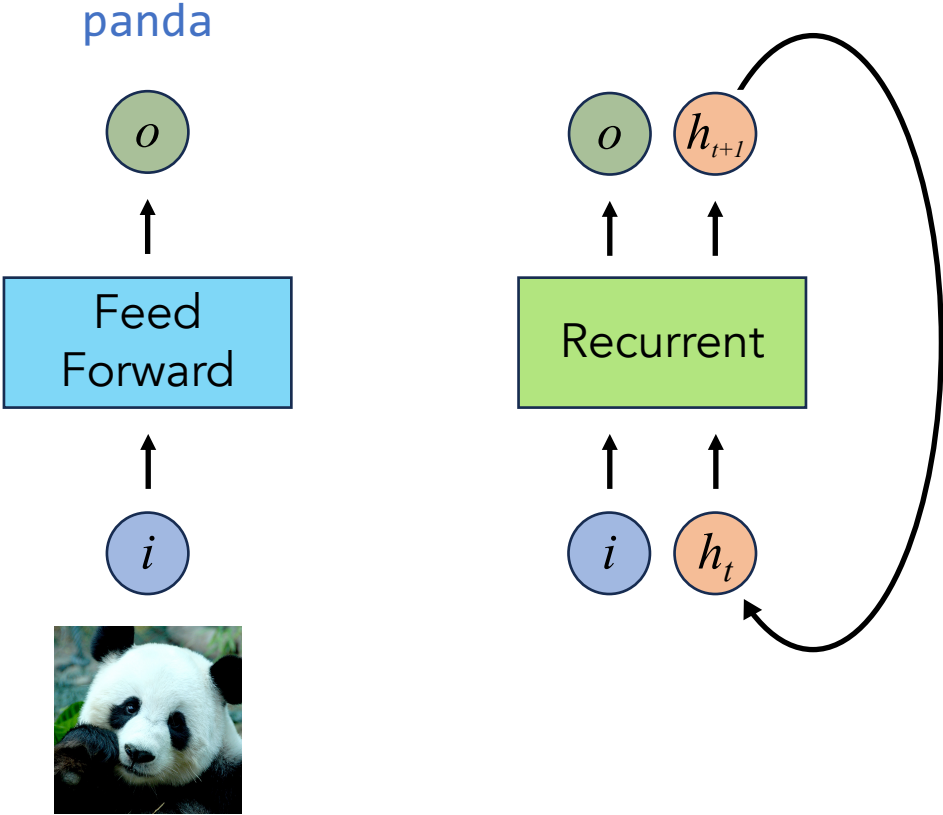
An Alternative Neural Network



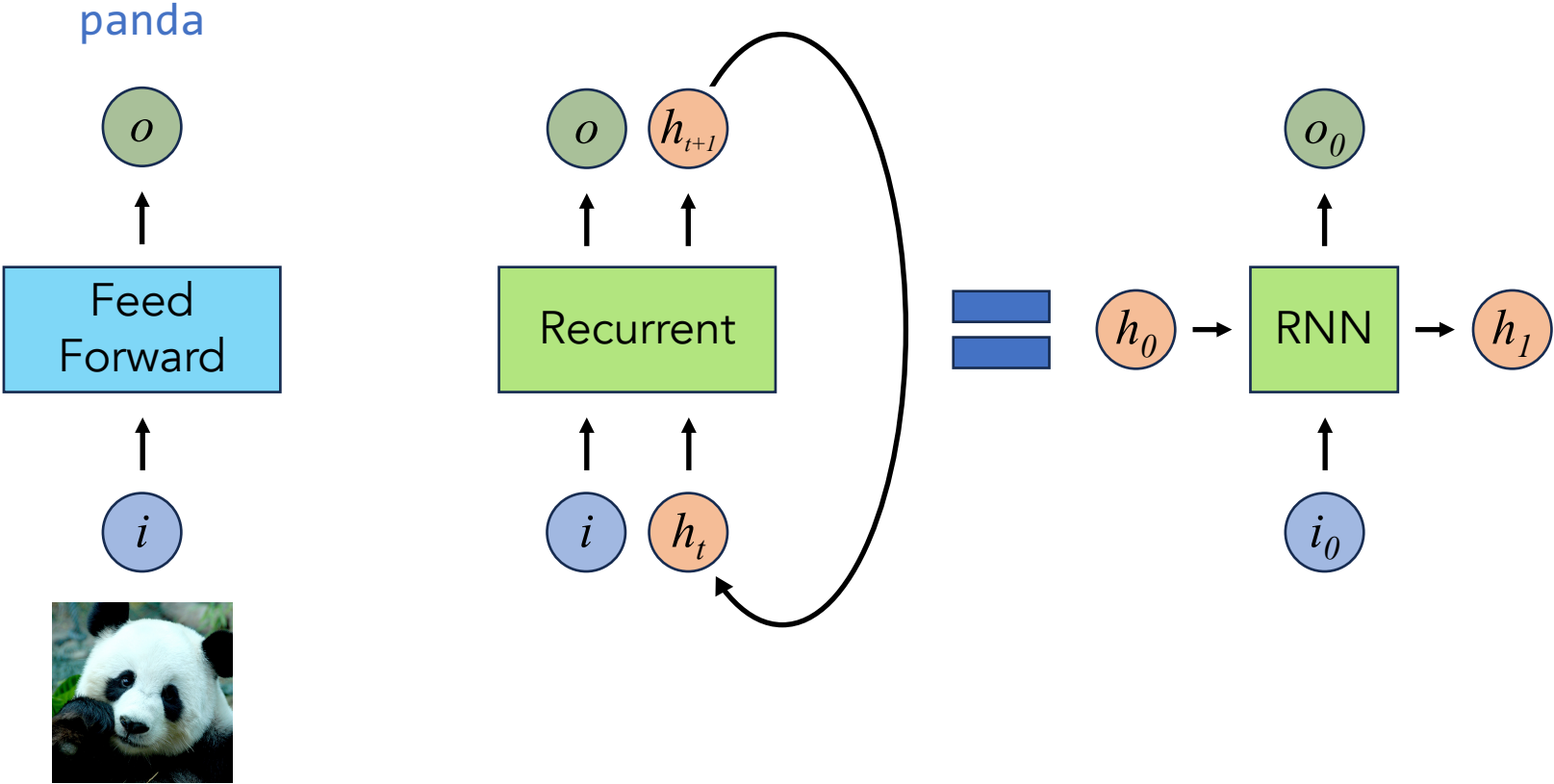
An Alternative Neural Network



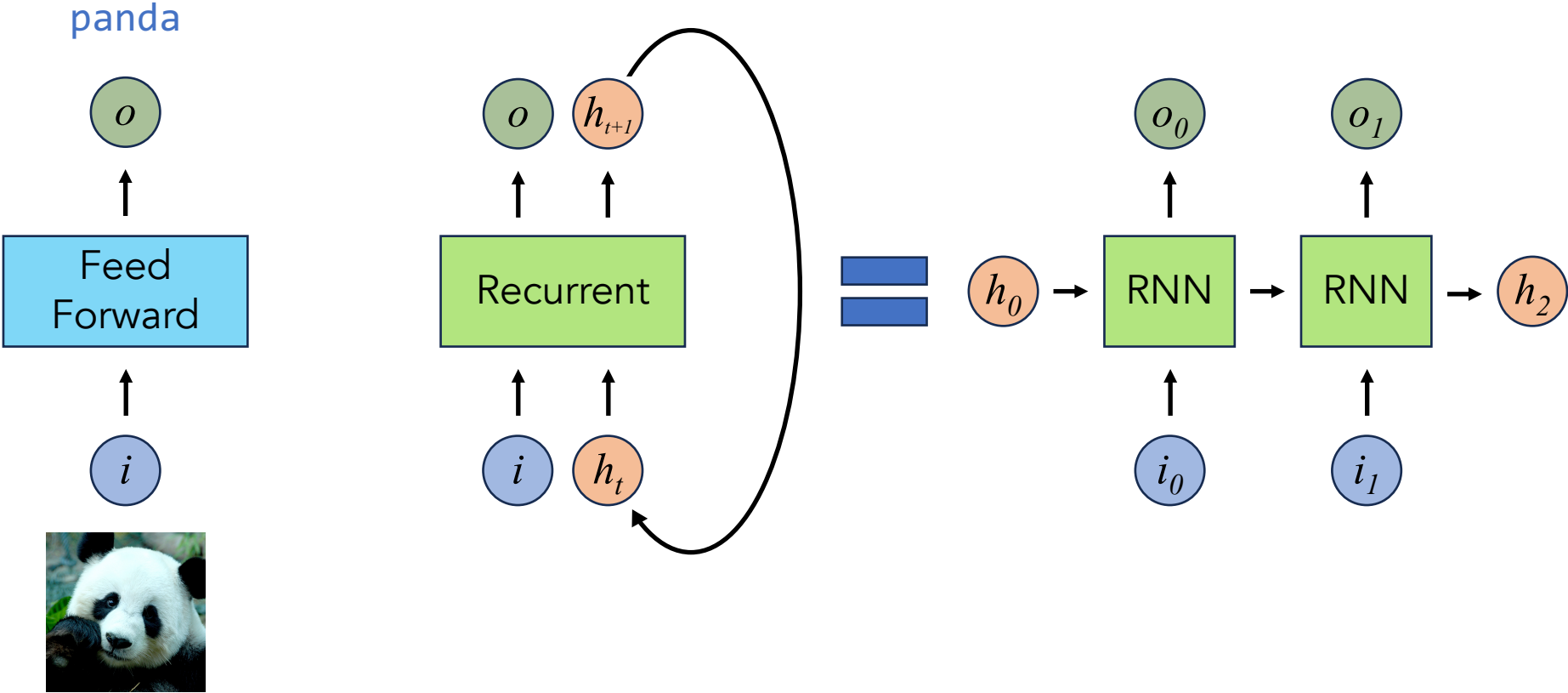
An Alternative Neural Network



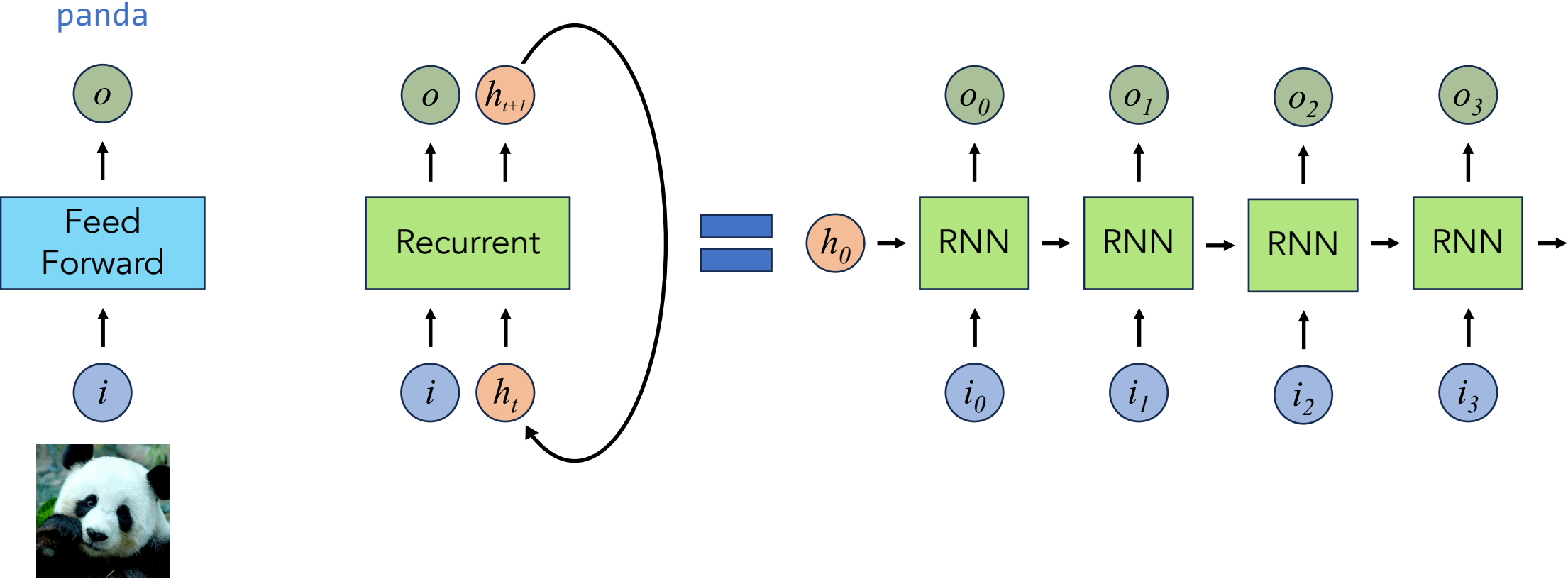
An Alternative Neural Network



An Alternative Neural Network



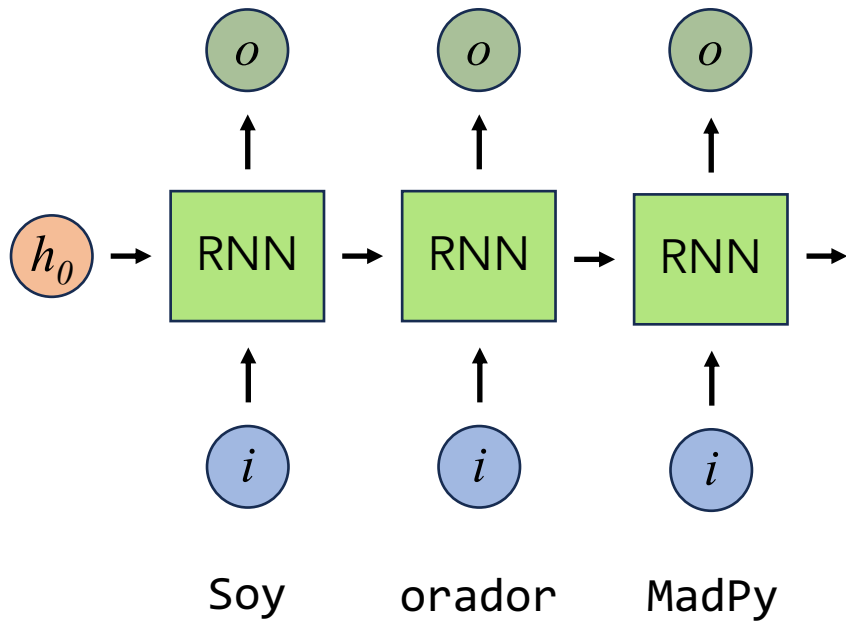
An Alternative Neural Network



RNNs for NLP

Soy orador MadPy

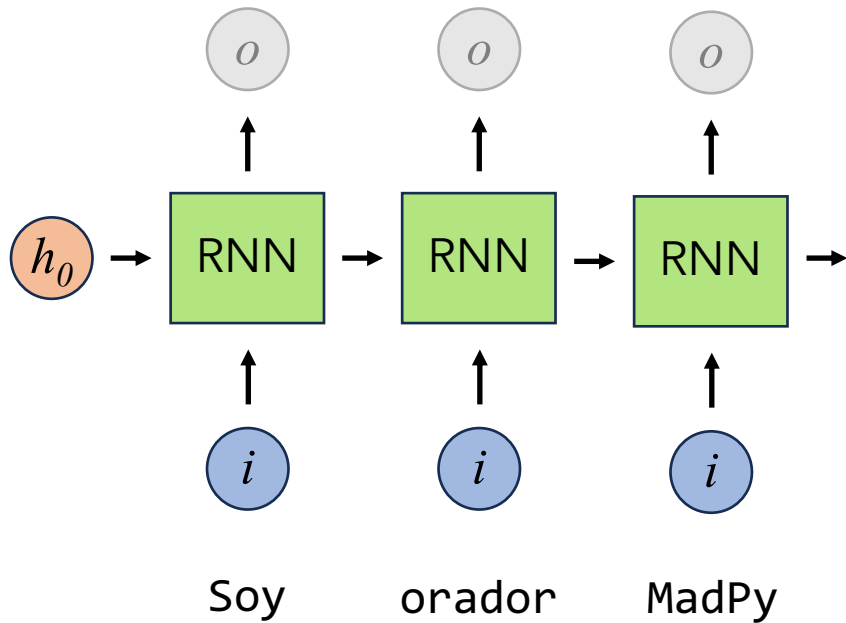
I am a MadPy speaker



RNNs for NLP

Soy orador MadPy

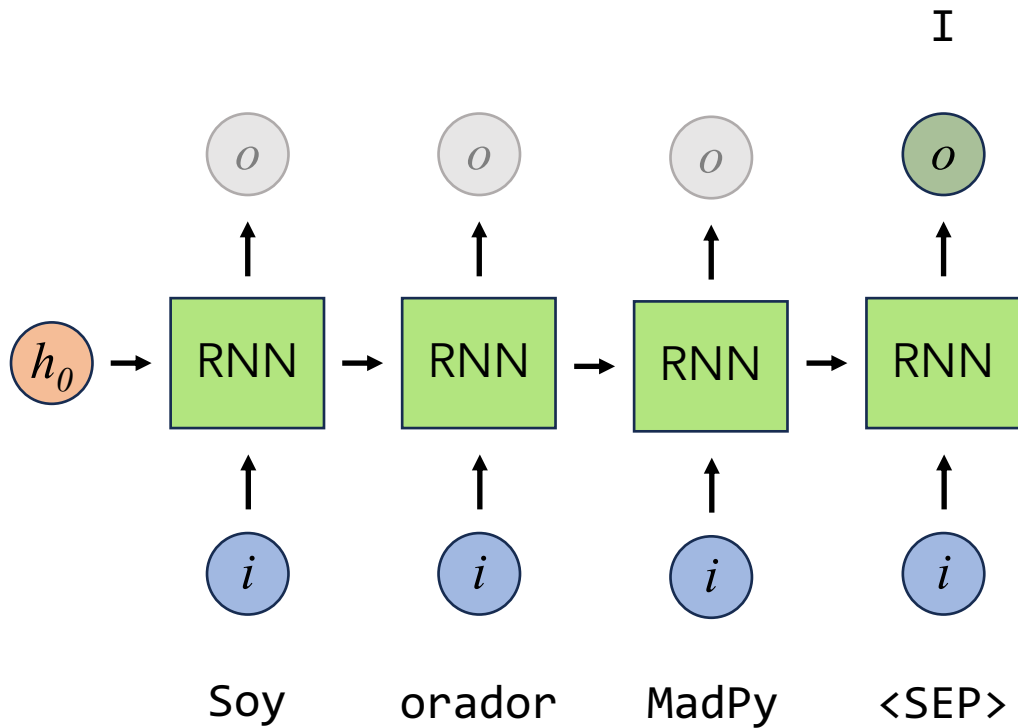
I am a MadPy speaker



RNNs for NLP

Soy orador MadPy

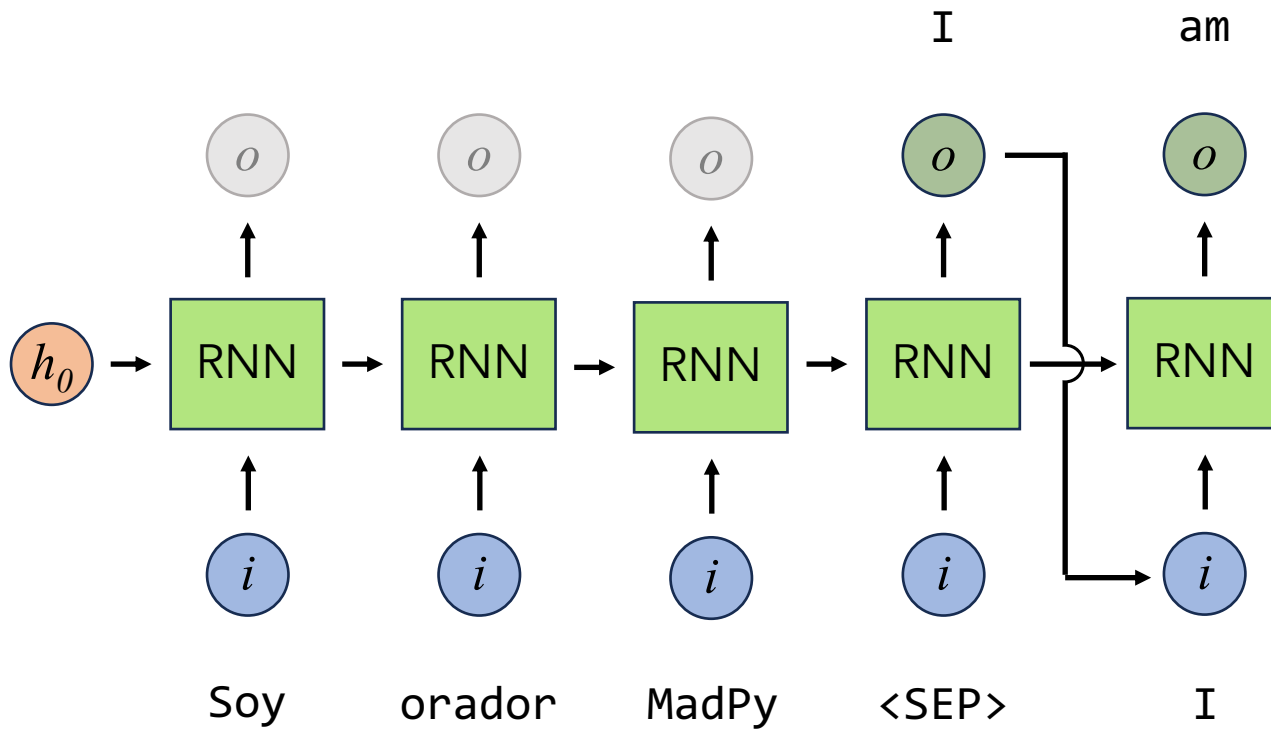
I am a MadPy speaker



RNNs for NLP

Soy orador MadPy

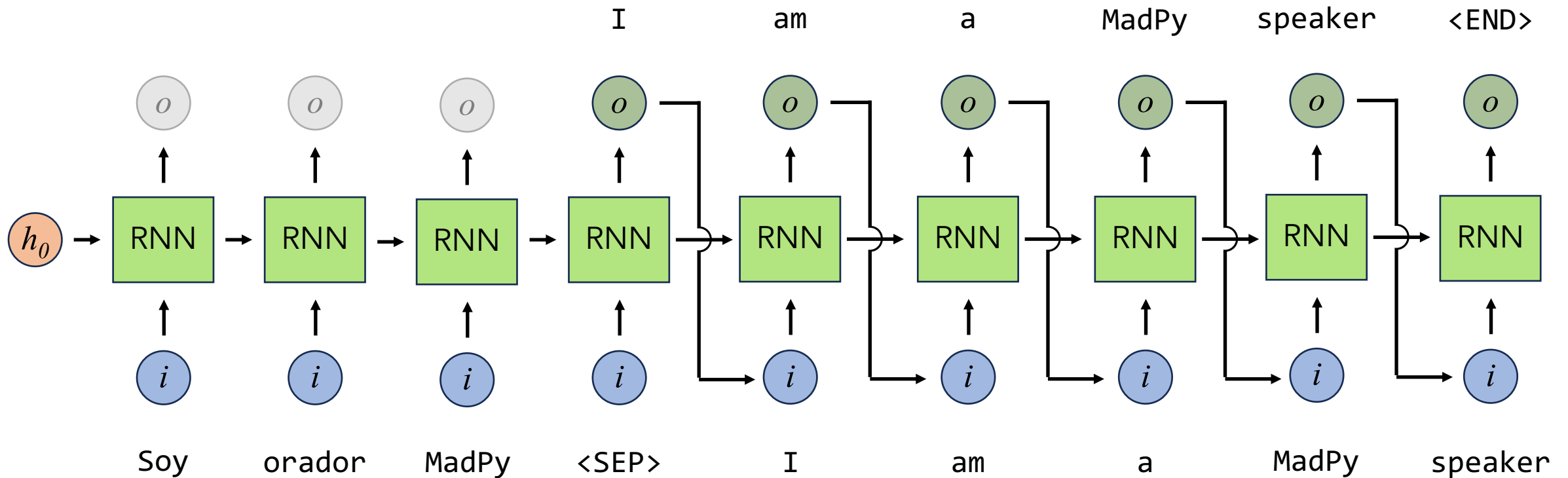
I am a MadPy speaker



RNNs for NLP

Soy orador MadPy

I am a MadPy speaker



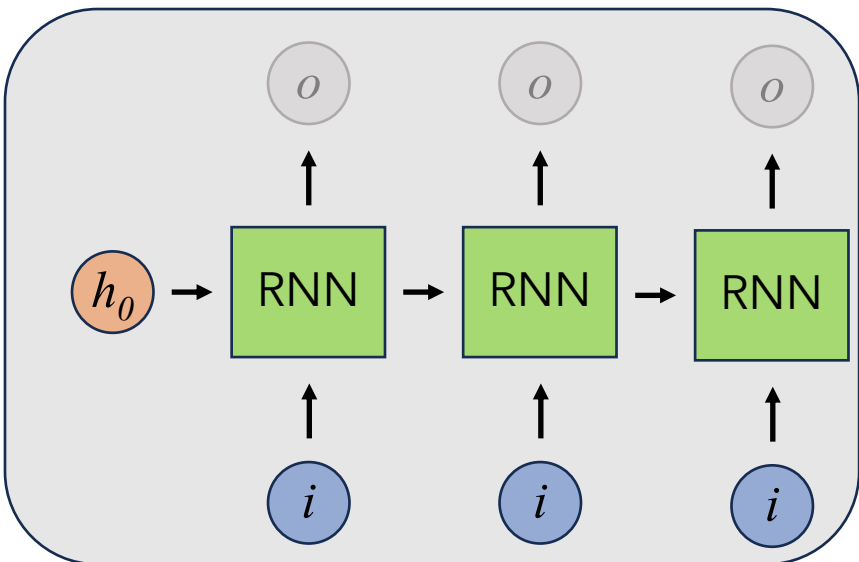
RNNs for NLP

Soy orador MadPy

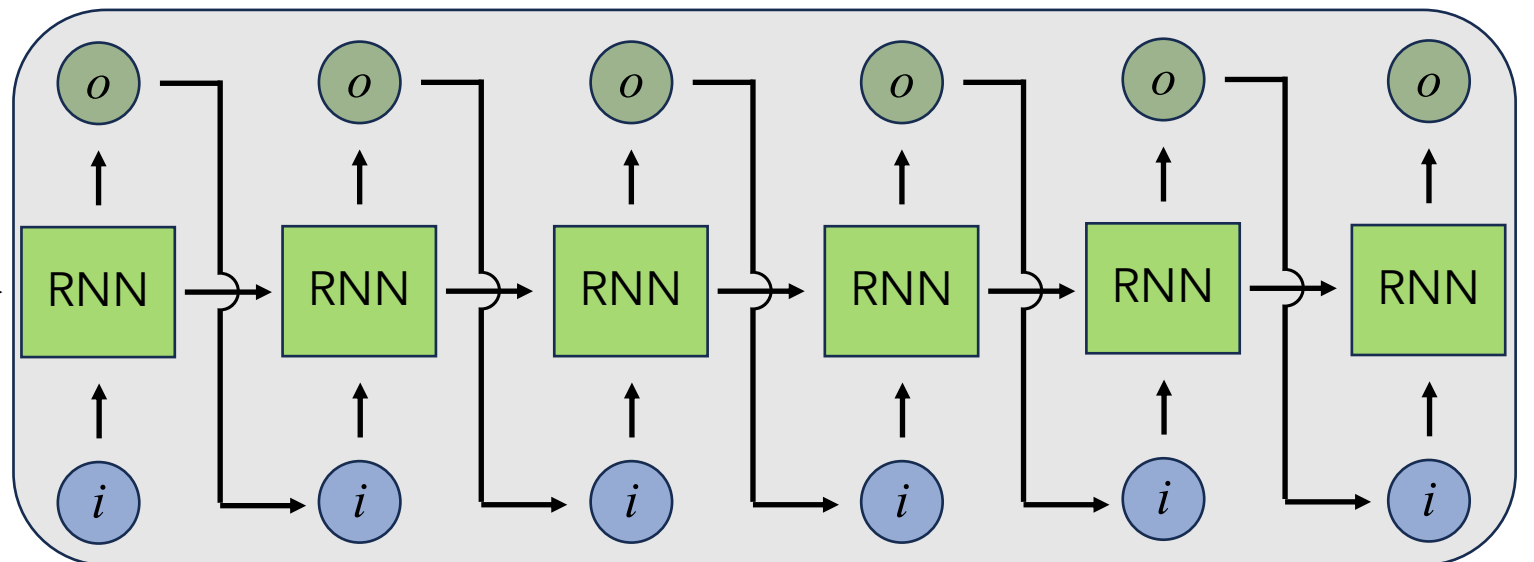
I am a MadPy speaker

ENCODER

DECODER

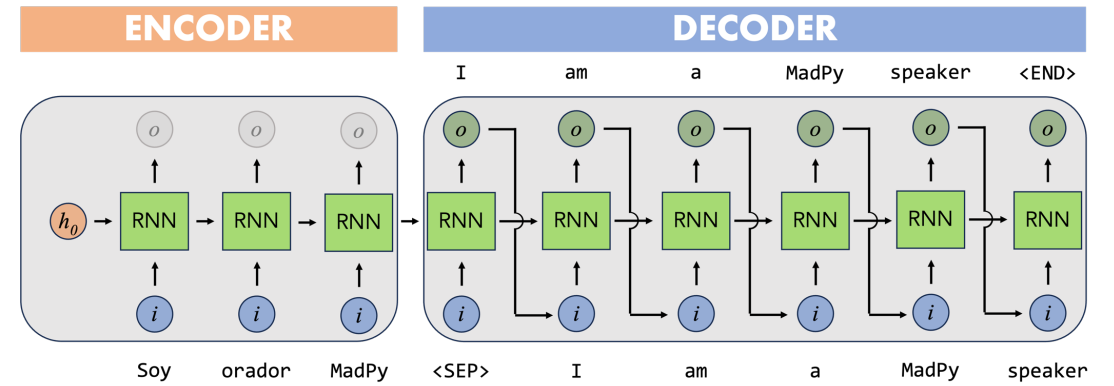


Soy orador MadPy



$\langle \text{SEP} \rangle$ I am a MadPy speaker

RNNs for NLP



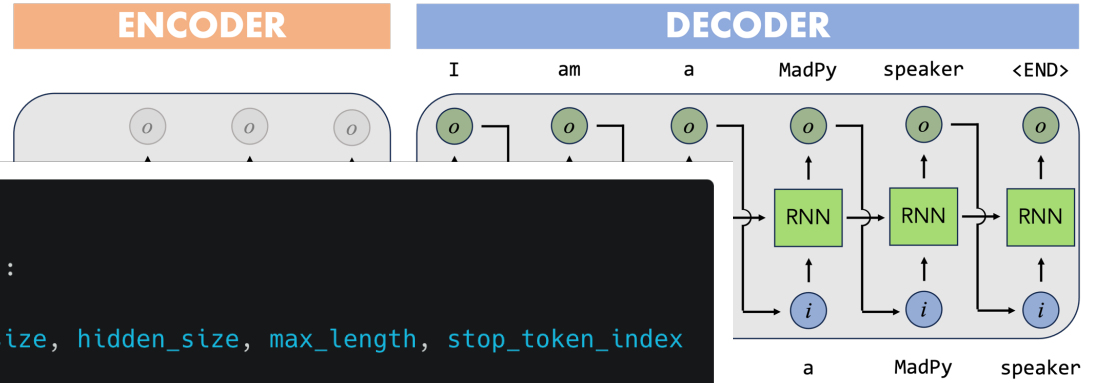
ENCODER

```
1 class EncoderRNN(nn.Module):
2     def __init__(self, input_size, embedding_size, hidden_size):
3         super(EncoderRNN, self).__init__()
4         self.hidden_size = hidden_size
5         self.embedding = nn.Embedding(input_size, embedding_size)
6         self.rnn = nn.RNN(embedding_size, hidden_size)
7
8     def forward(self, input, hidden=None):
9         embedded = self.embedding(input)
10        output, hidden = self.rnn(embedded, hidden)
11        return output, hidden
```

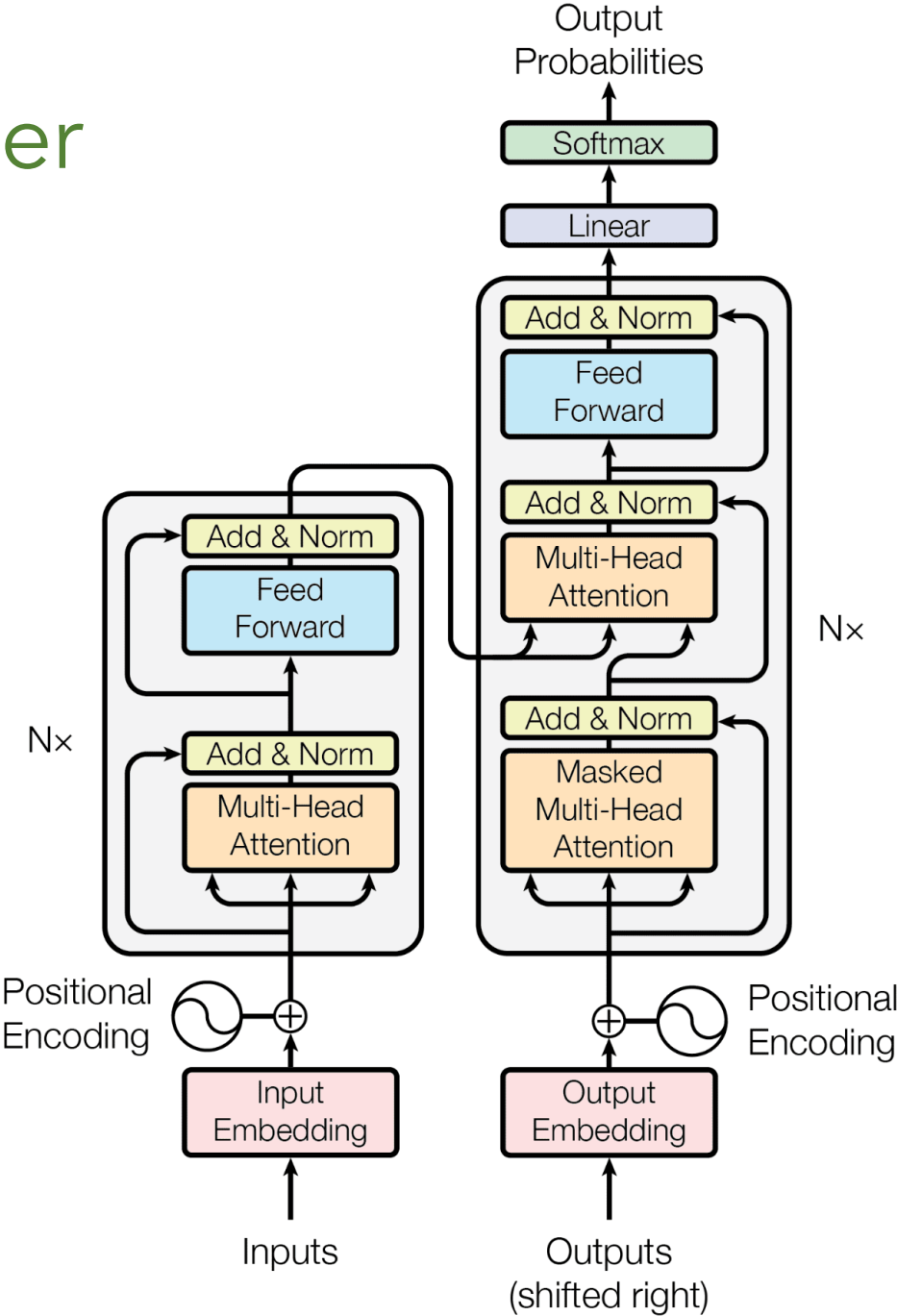
RNNs for NLP

DECODER

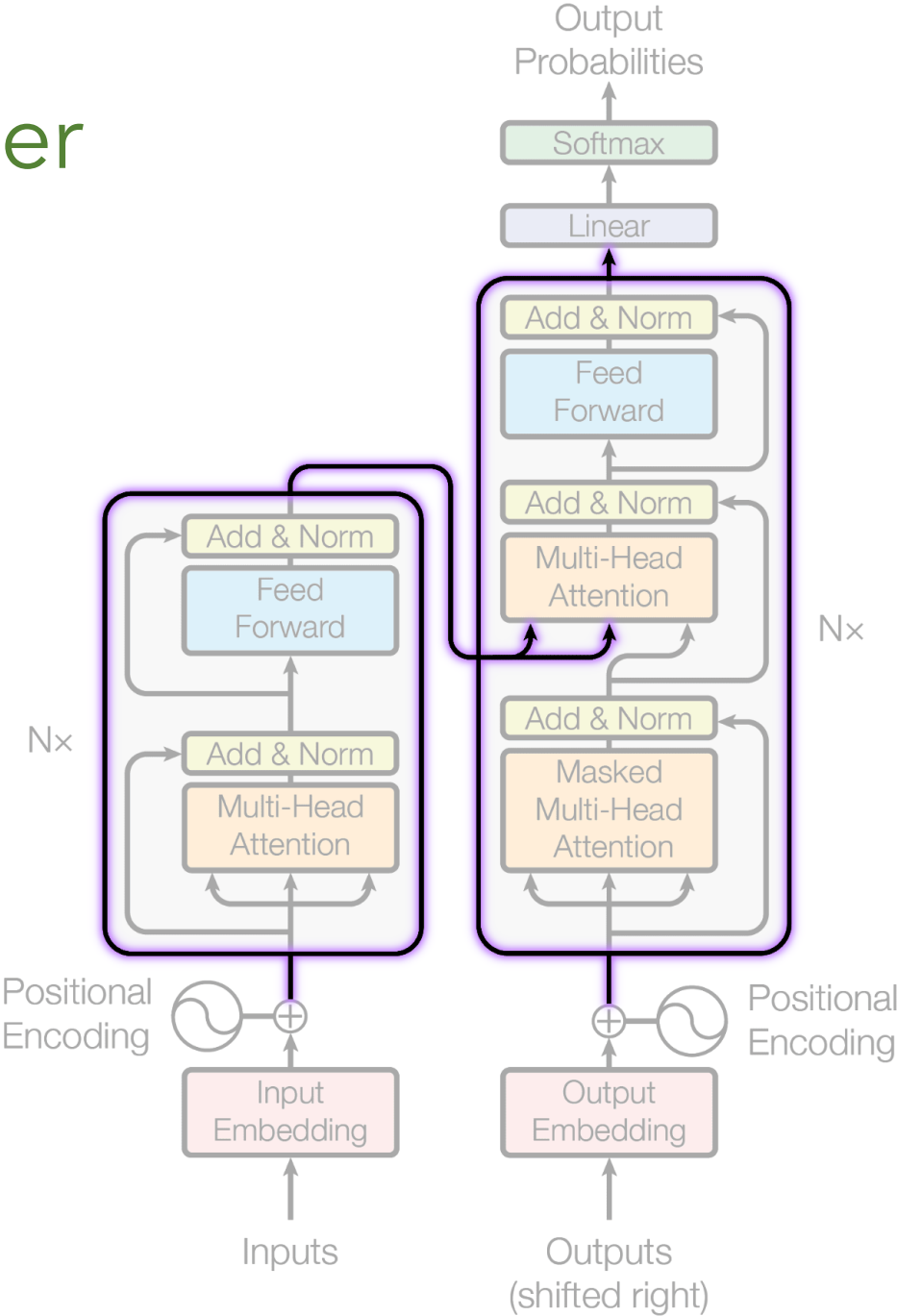
```
1 class AutoRegressiveDecoder(nn.Module):
2     def __init__(
3         self, output_size, embedding_size, hidden_size, max_length, stop_token_index
4     ):
5         super(AutoRegressiveDecoder, self).__init__()
6         self.hidden_size = hidden_size
7         self.embedding = nn.Embedding(output_size, embedding_size)
8         self.rnn = nn.RNN(embedding_size, hidden_size)
9         self.out = nn.Linear(hidden_size, output_size)
10        self.max_length = max_length
11        self.stop_token_index = stop_token_index
12
13    def forward(self, hidden, start_token_index):
14        device = hidden.device
15        input = torch.tensor([[start_token_index]], device=device)
16        outputs = []
17        logits_list = []
18        for _ in range(self.max_length):
19            embedded = self.embedding(input)
20            rnn_output, hidden = self.rnn(embedded, hidden)
21            logits = self.out(rnn_output[0])
22            logits_list.append(logits)
23            top1 = logits.argmax(1)
24            outputs.append(top1.item())
25            if top1.item() == self.stop_token_index:
26                break
27            input = top1.unsqueeze(0)
28        return outputs, torch.stack(logits_list, dim=0)
```



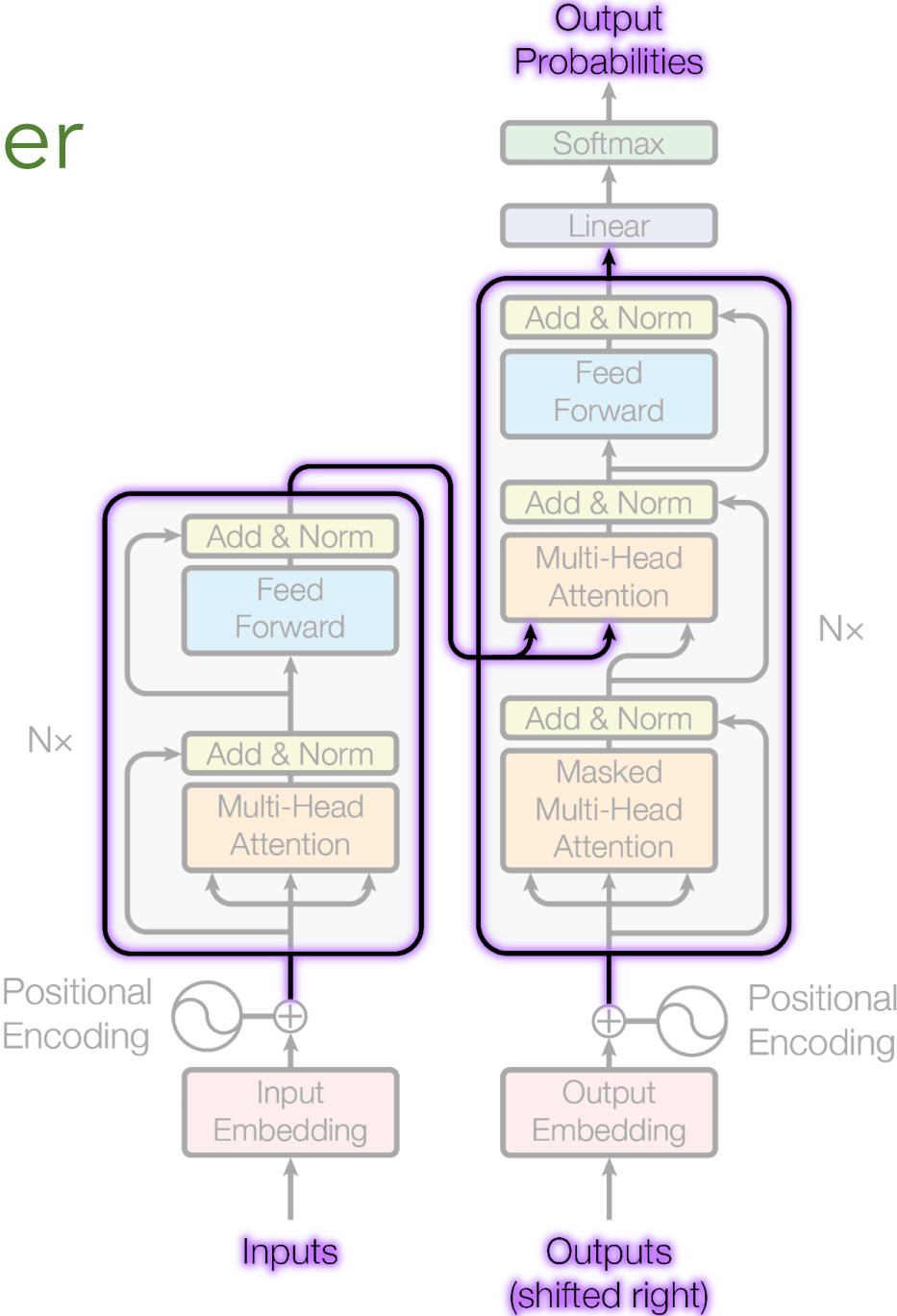
The Transformer



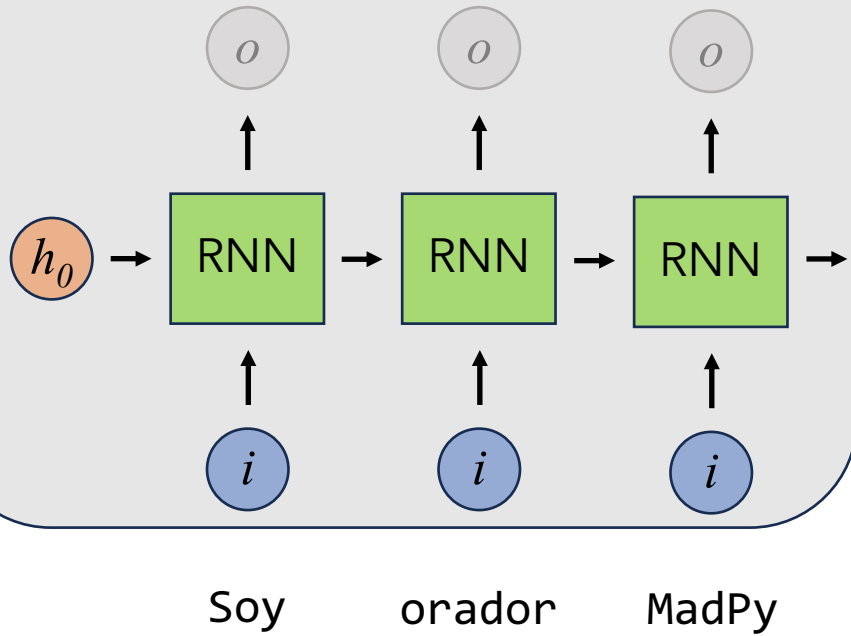
The Transformer



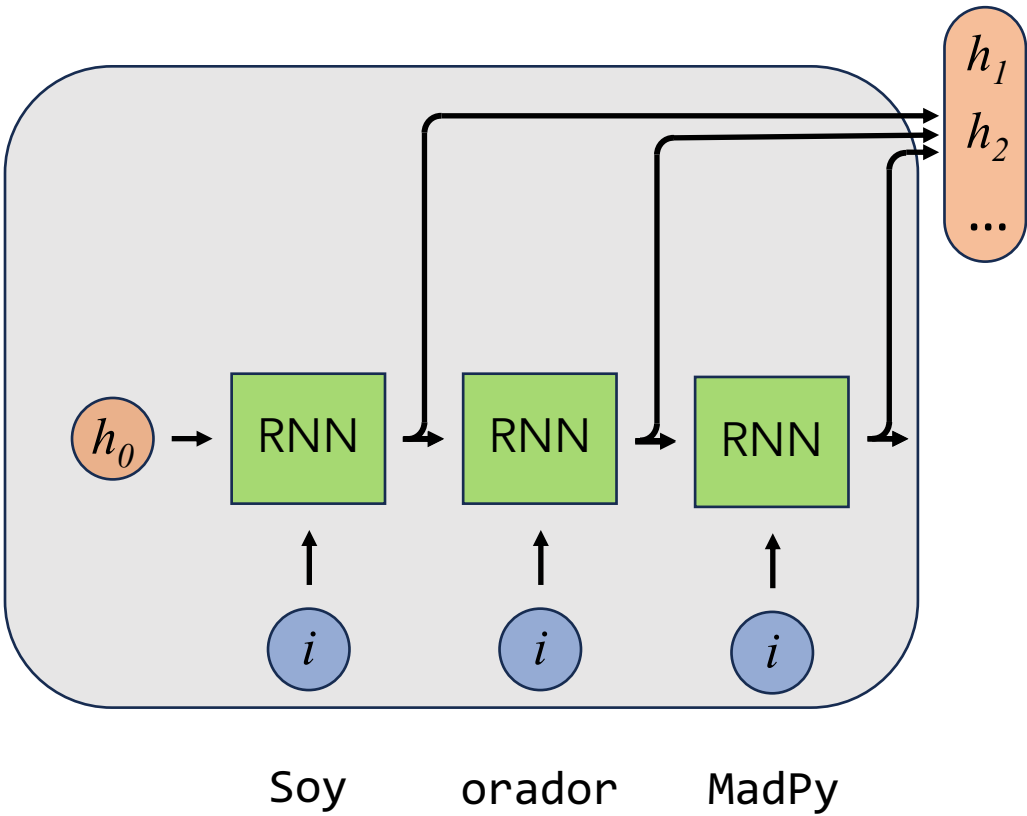
The Transformer



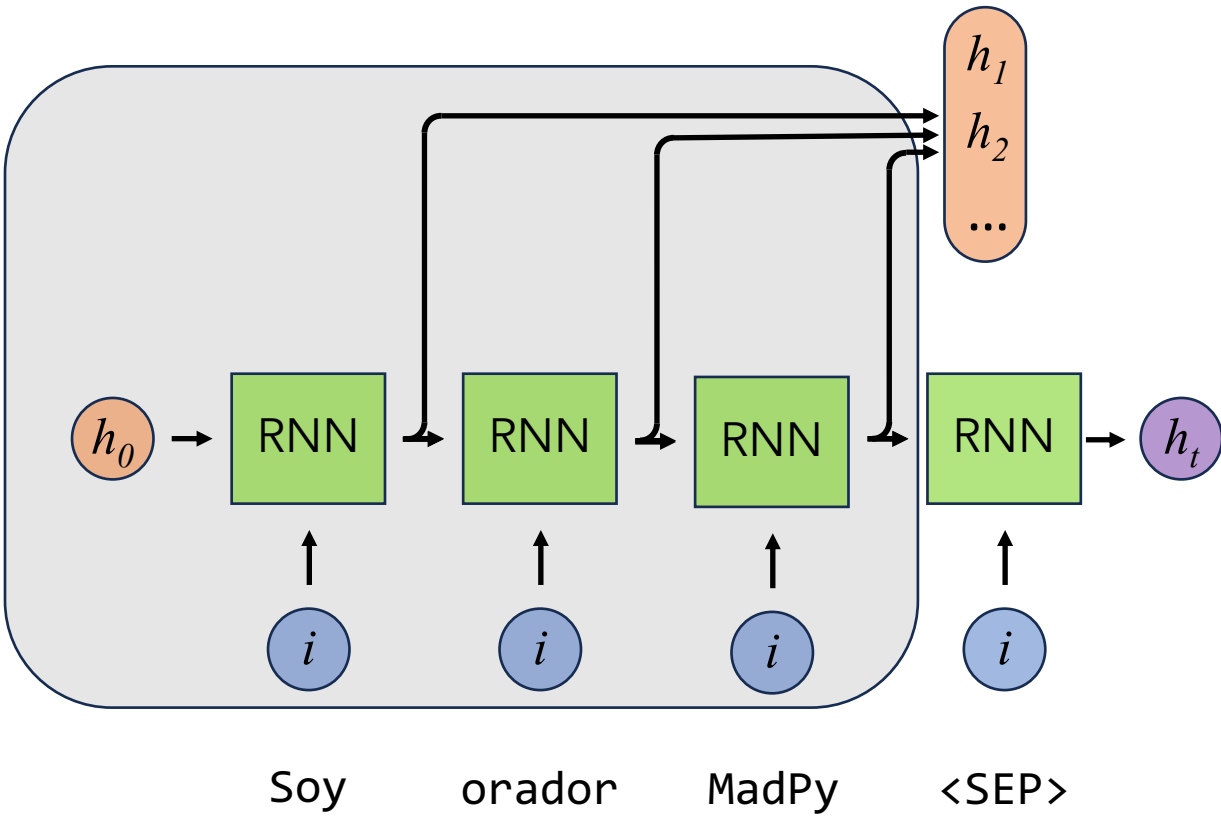
RNNsearch



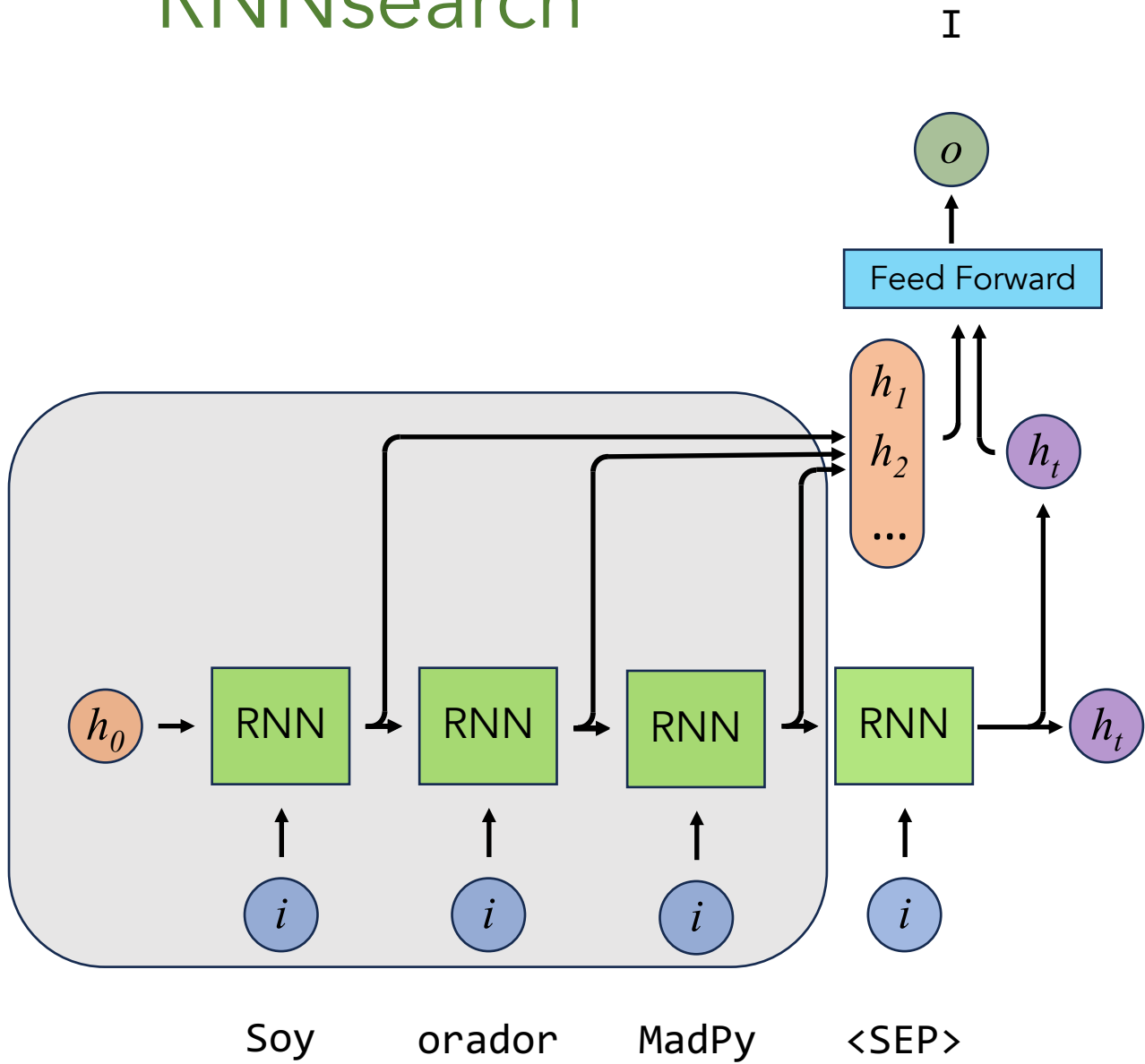
RNNsearch



RNNsearch

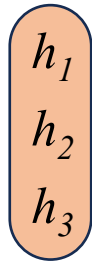


RNNsearch



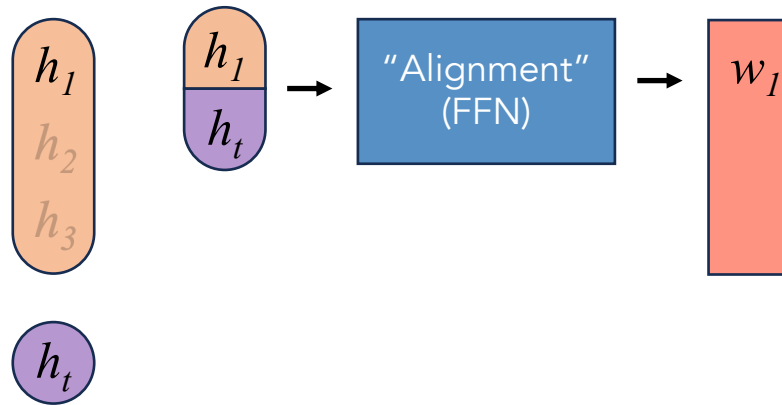
RNNsearch

The "Attention" Mechanism



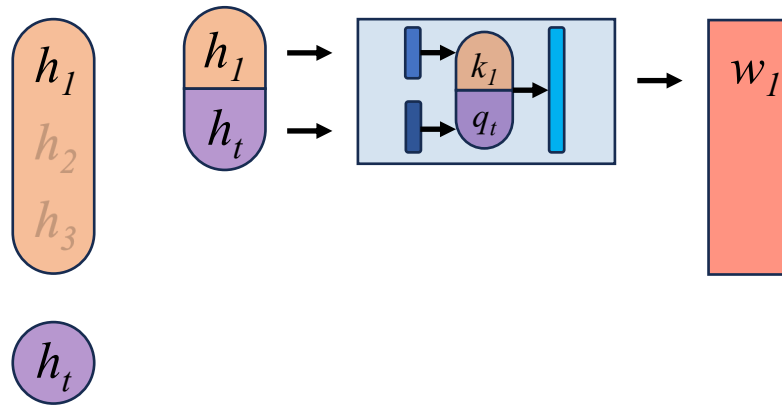
RNNsearch

The "Attention" Mechanism



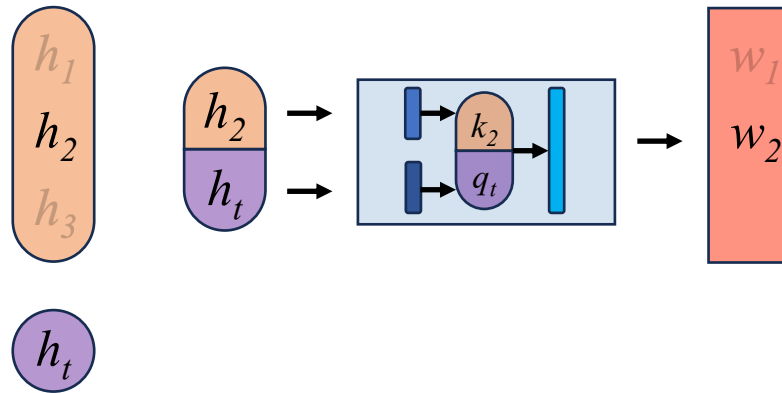
RNNsearch

The "Attention" Mechanism



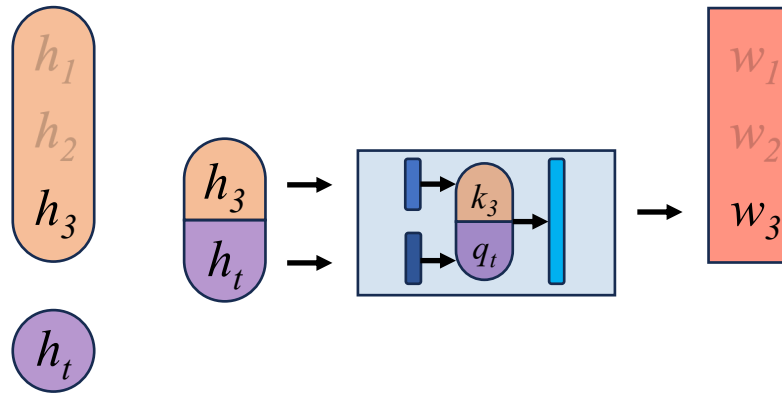
RNNsearch

The "Attention" Mechanism



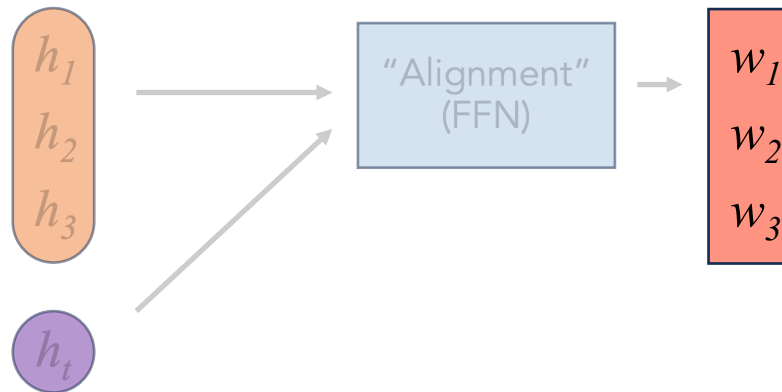
RNNsearch

The "Attention" Mechanism



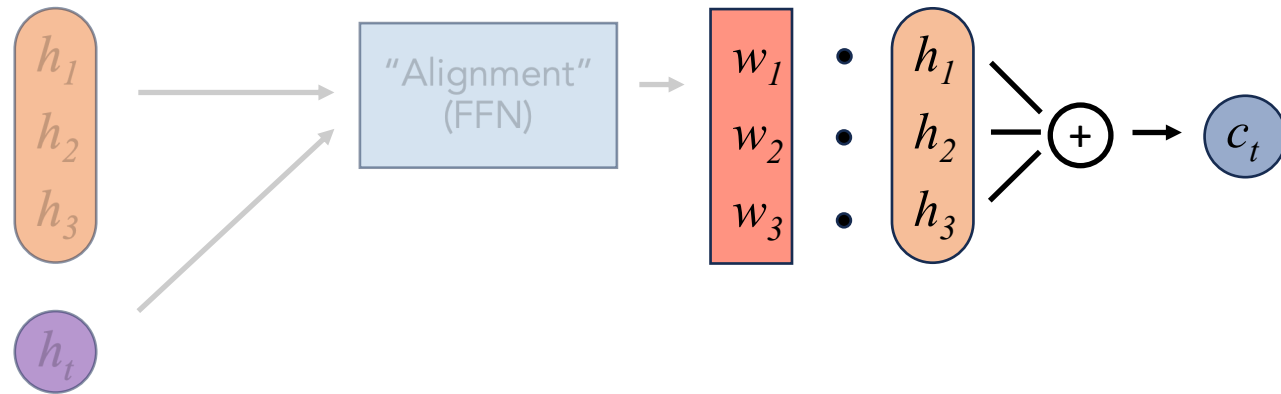
RNNsearch

The "Attention" Mechanism



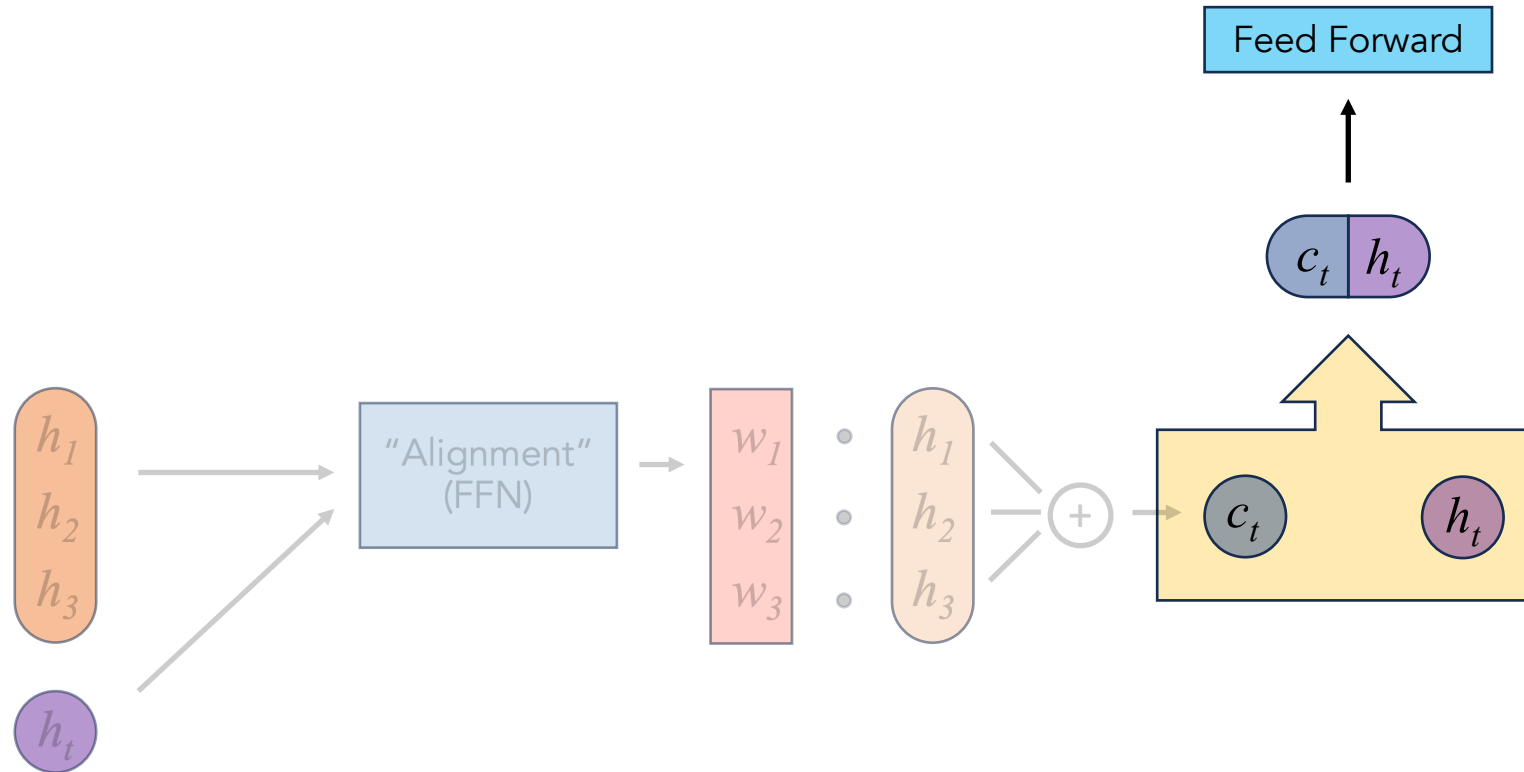
RNNsearch

The "Attention" Mechanism



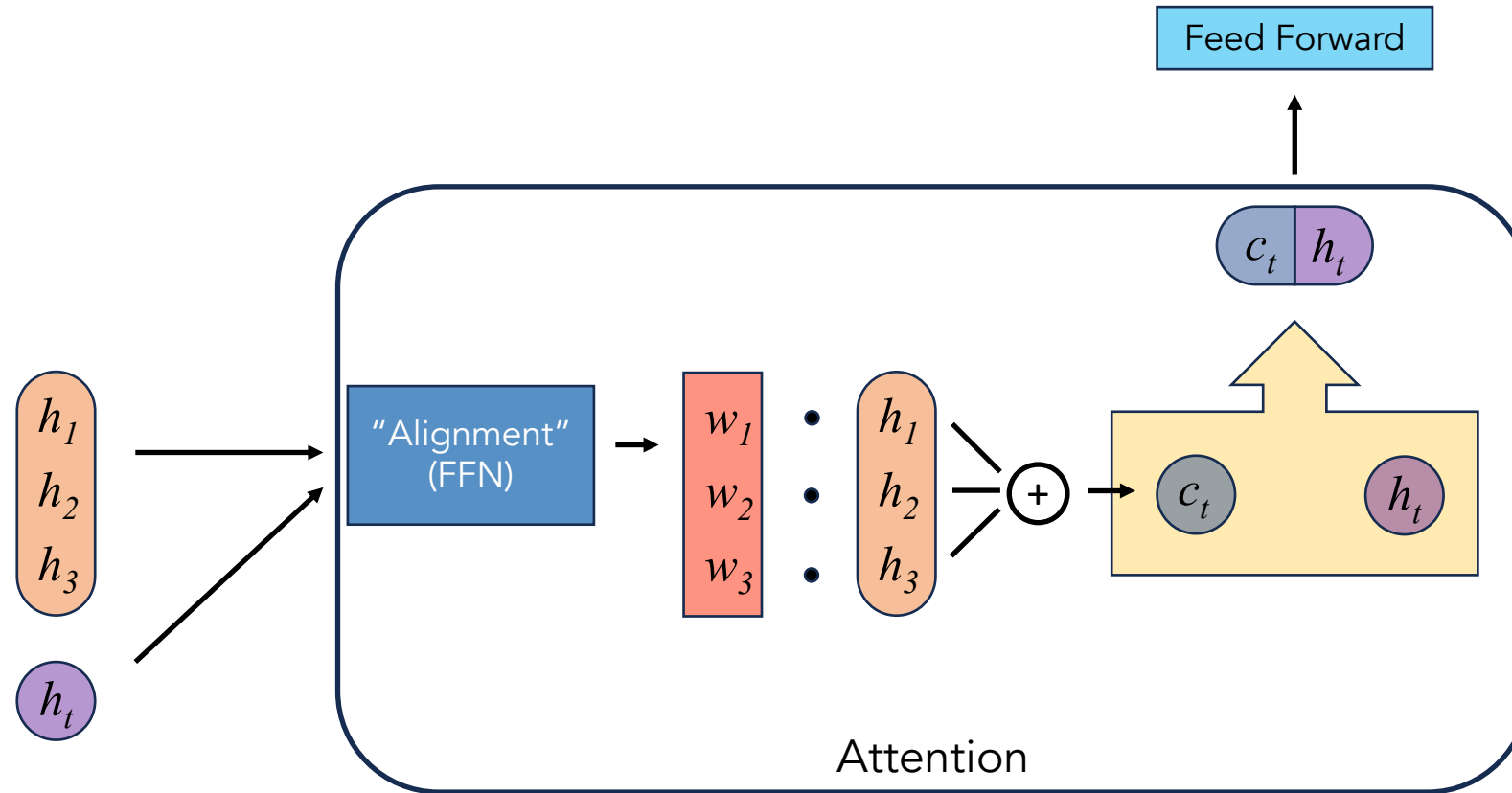
RNNsearch

The "Attention" Mechanism

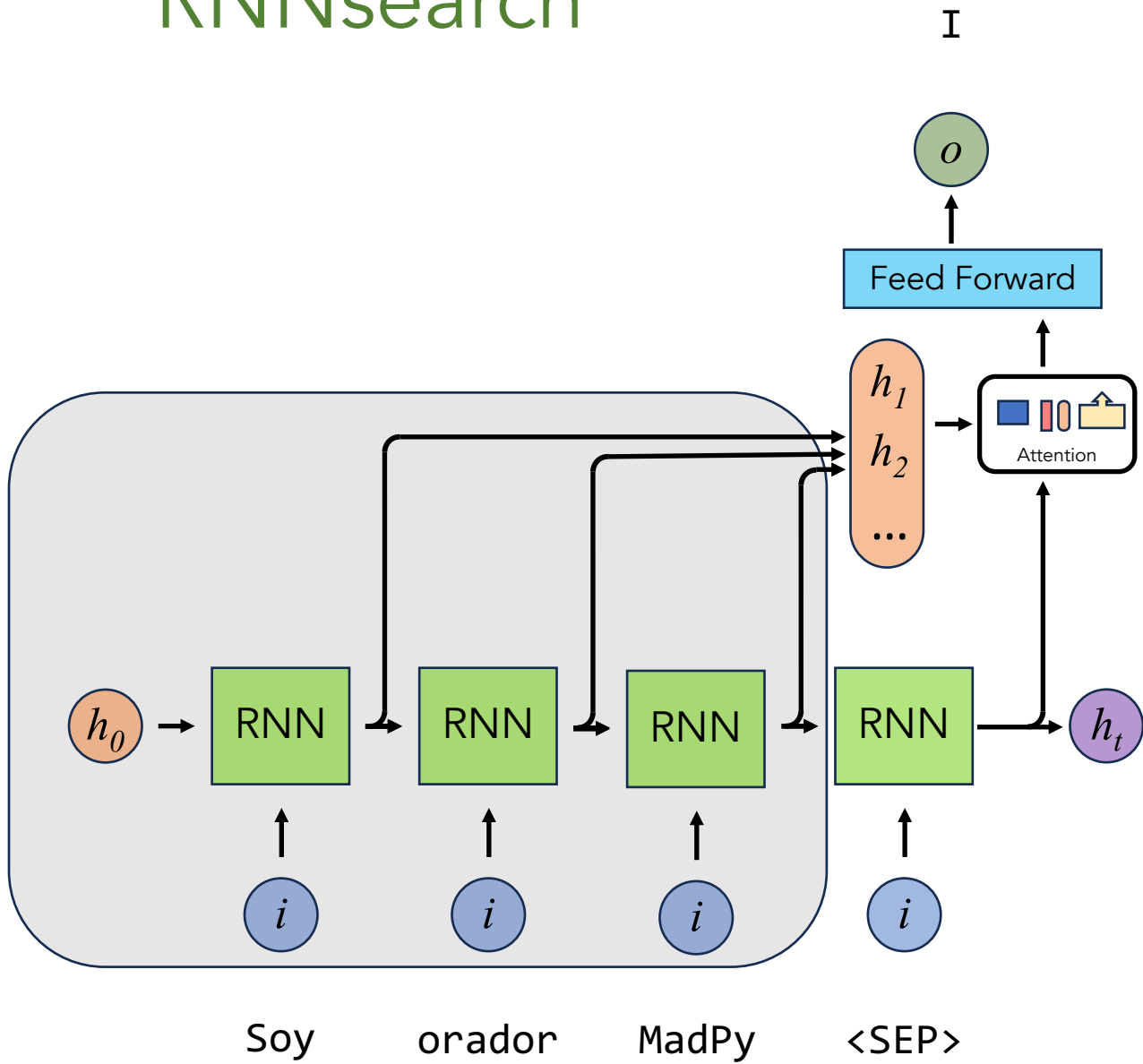


RNNsearch

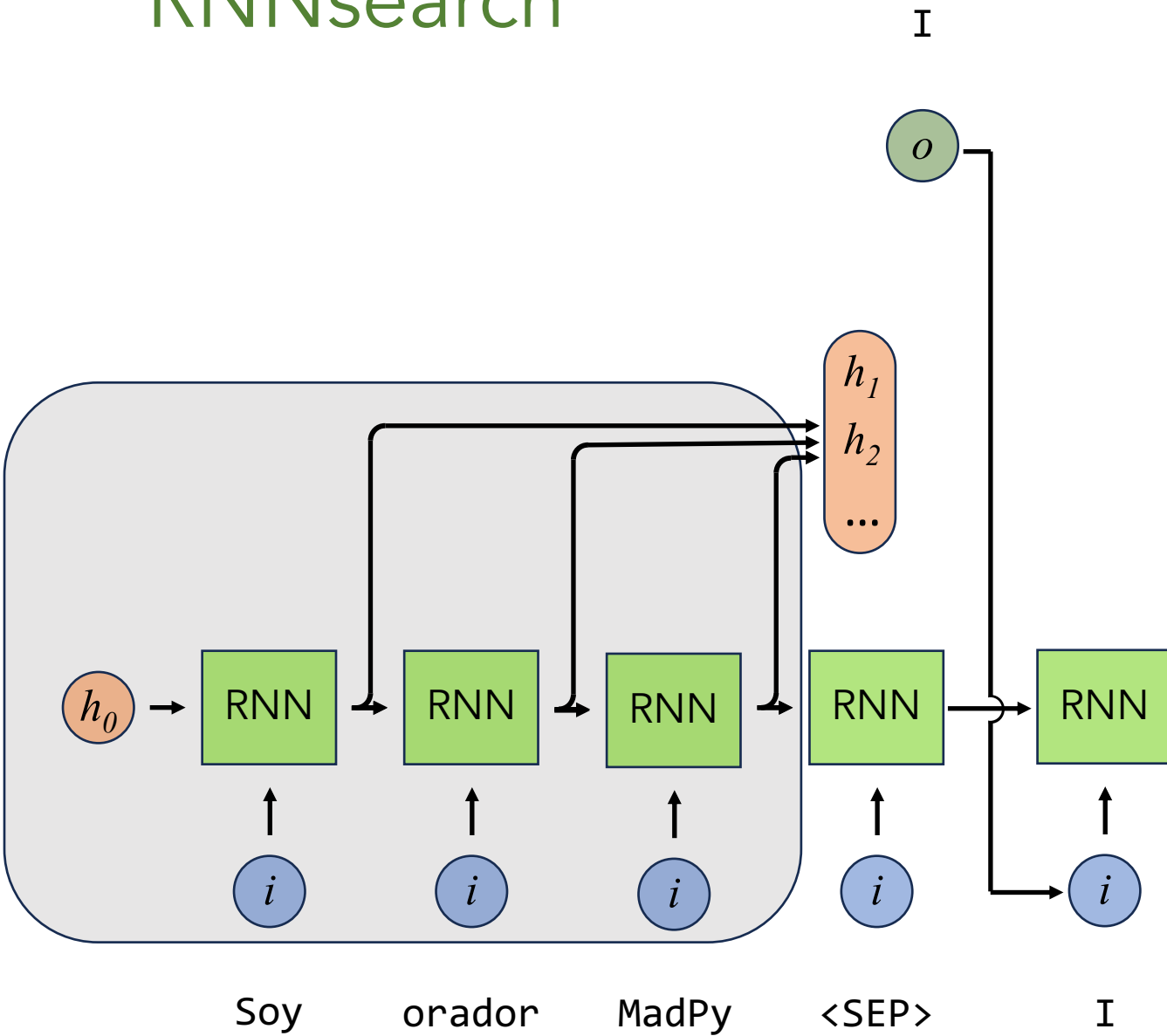
The "Attention" Mechanism



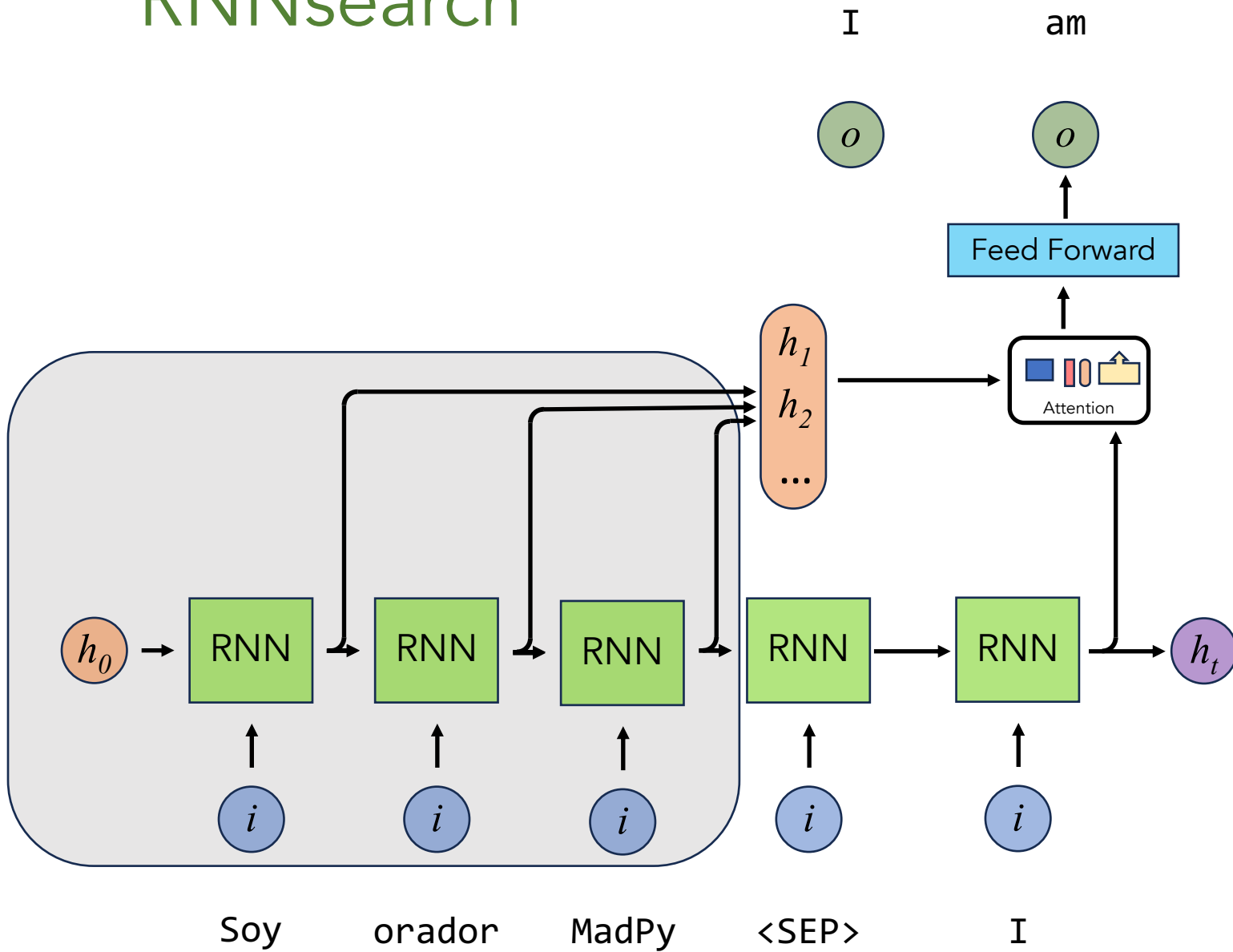
RNNsearch



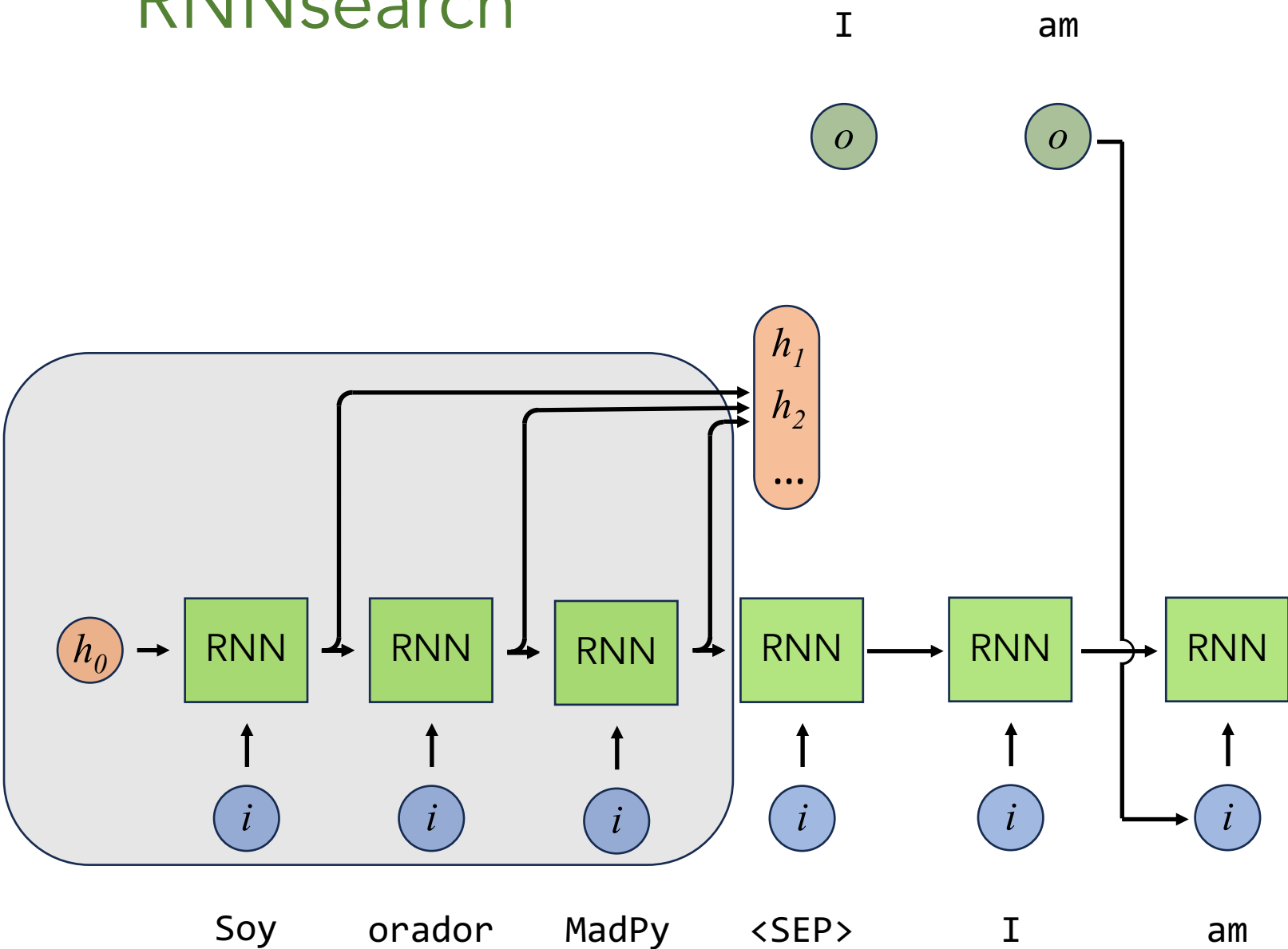
RNNsearch



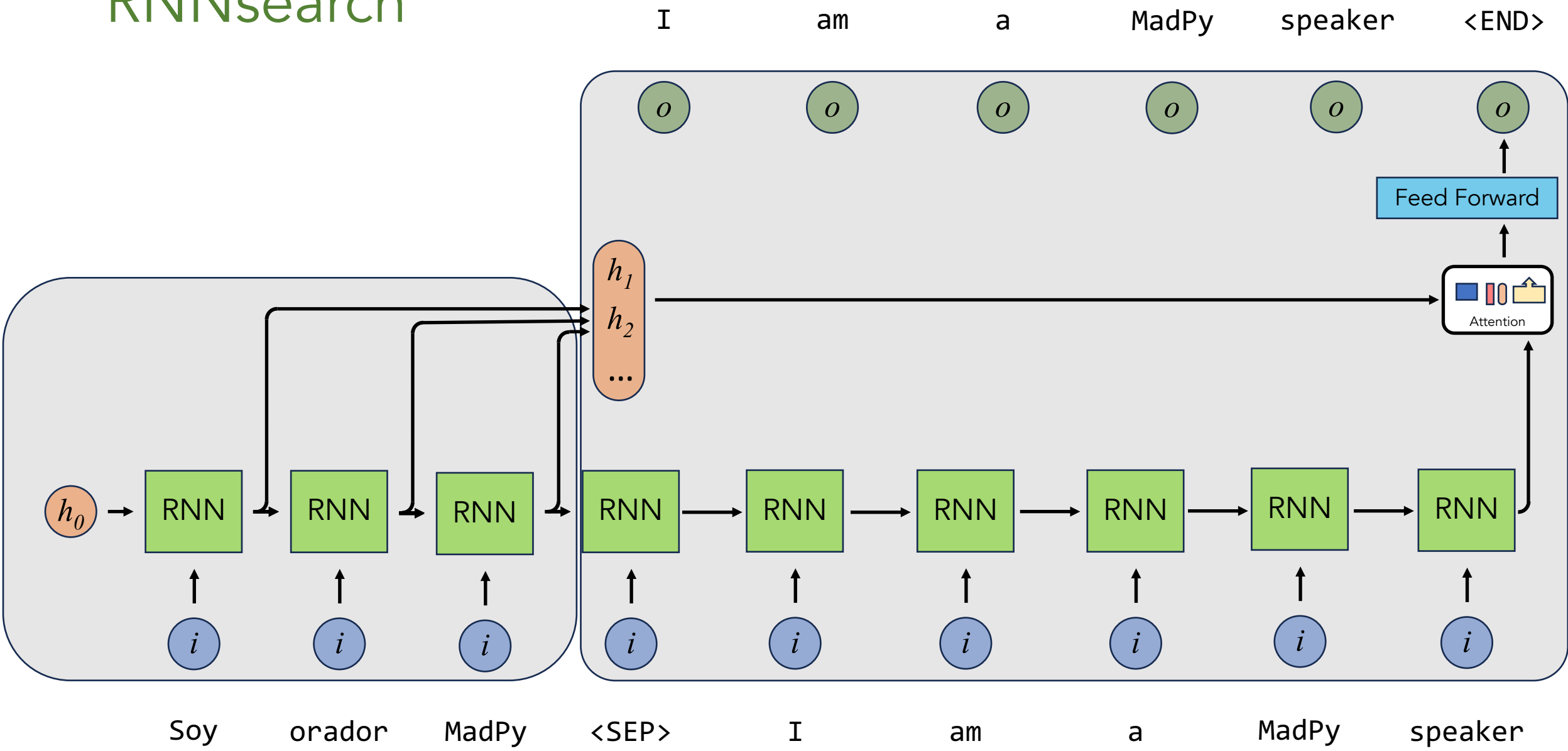
RNNsearch



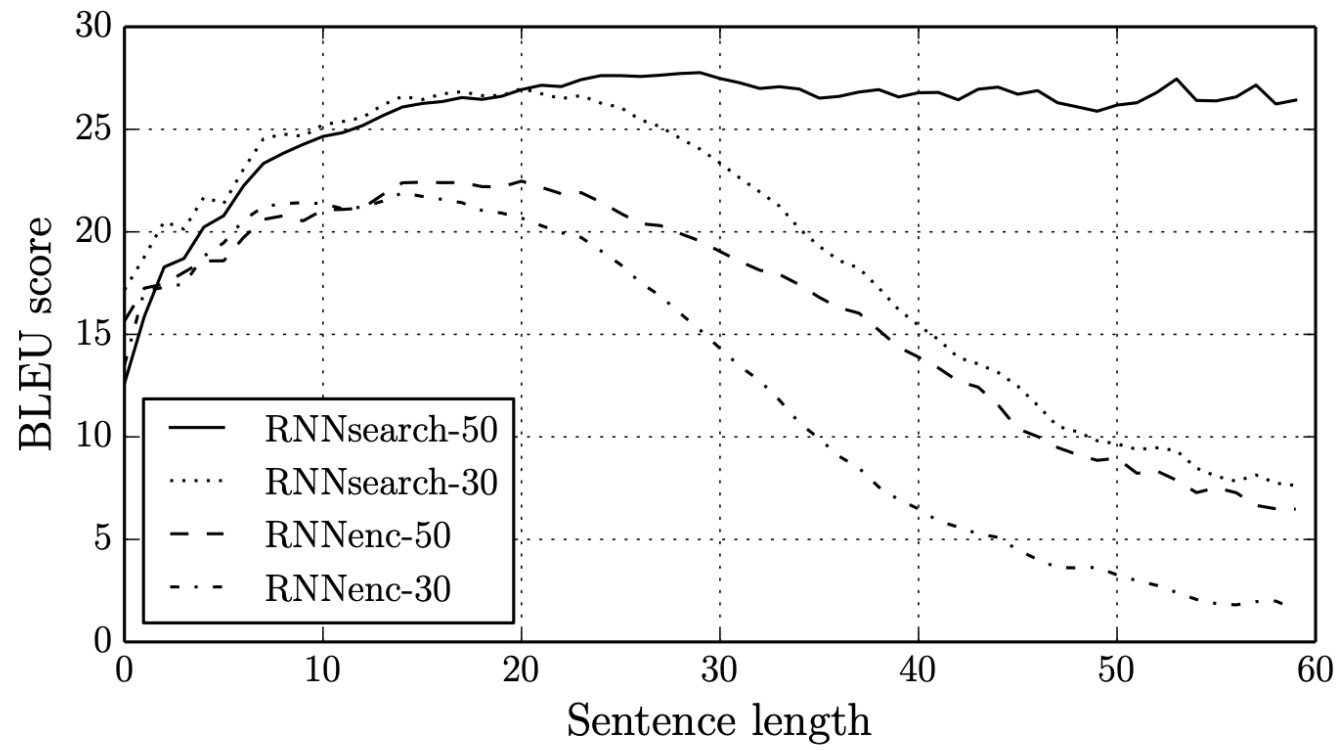
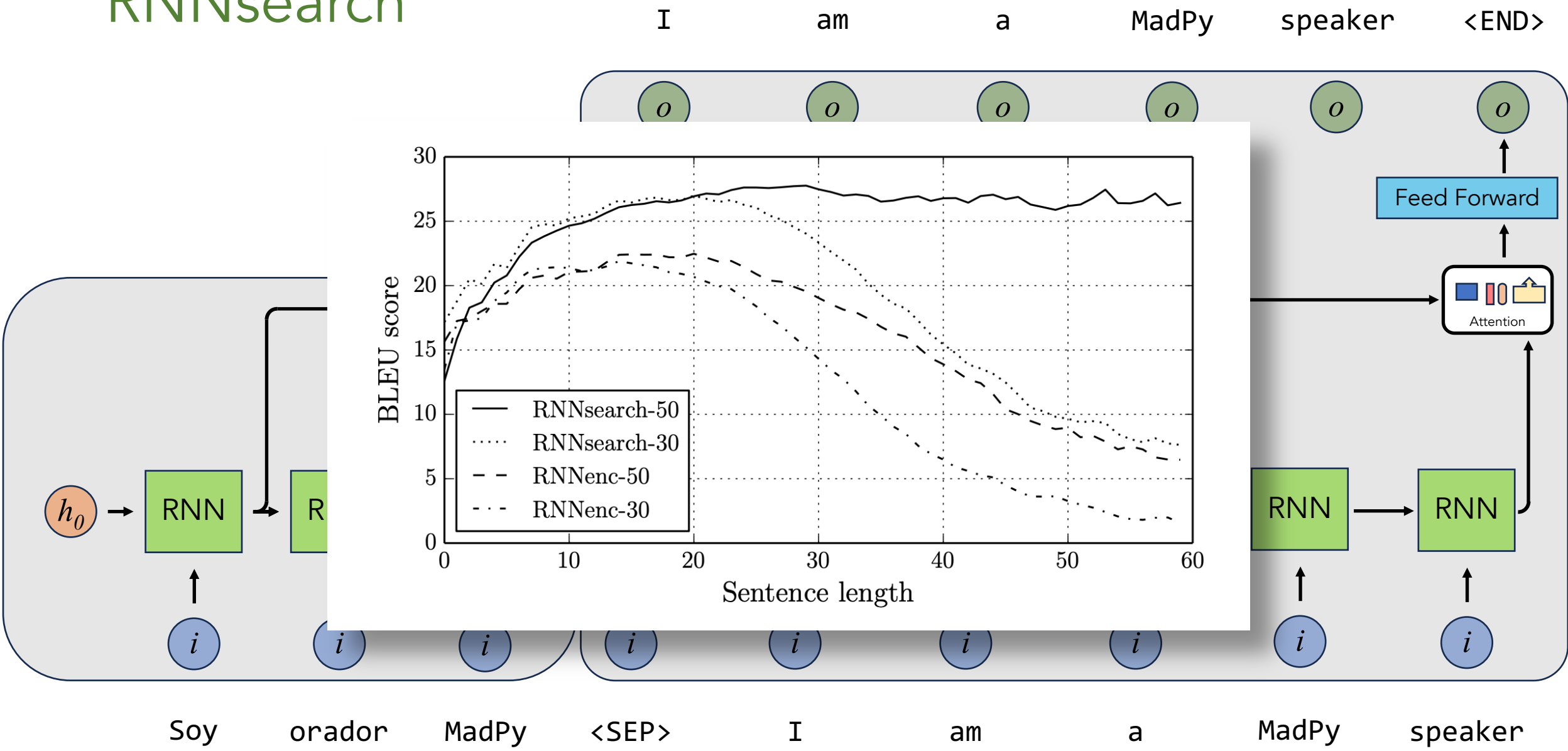
RNNsearch



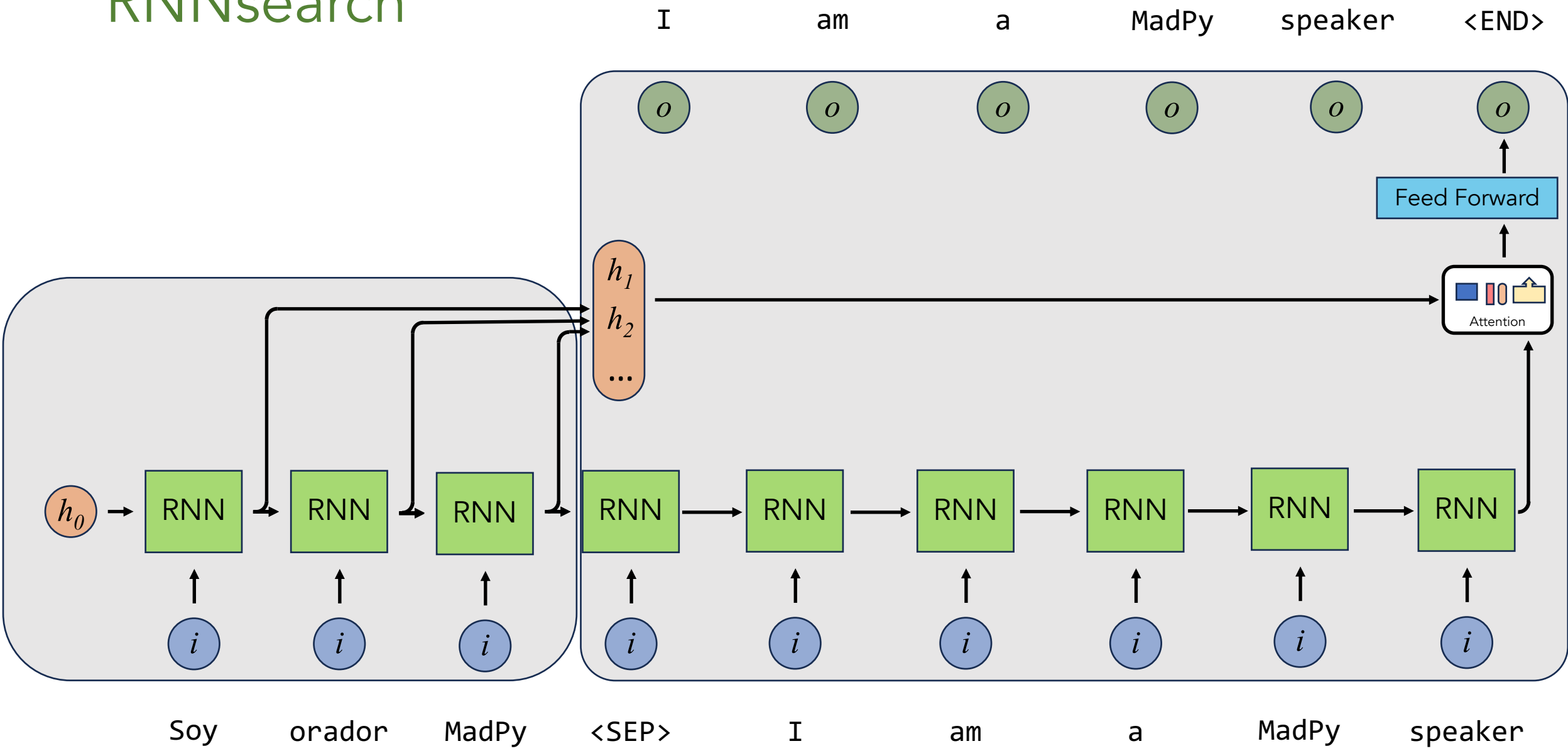
RNNsearch



RNNsearch



RNNsearch



[cs.CL] 12 Jun 2017

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez*[†]
University of Toronto
aidan@cs.toronto.edu

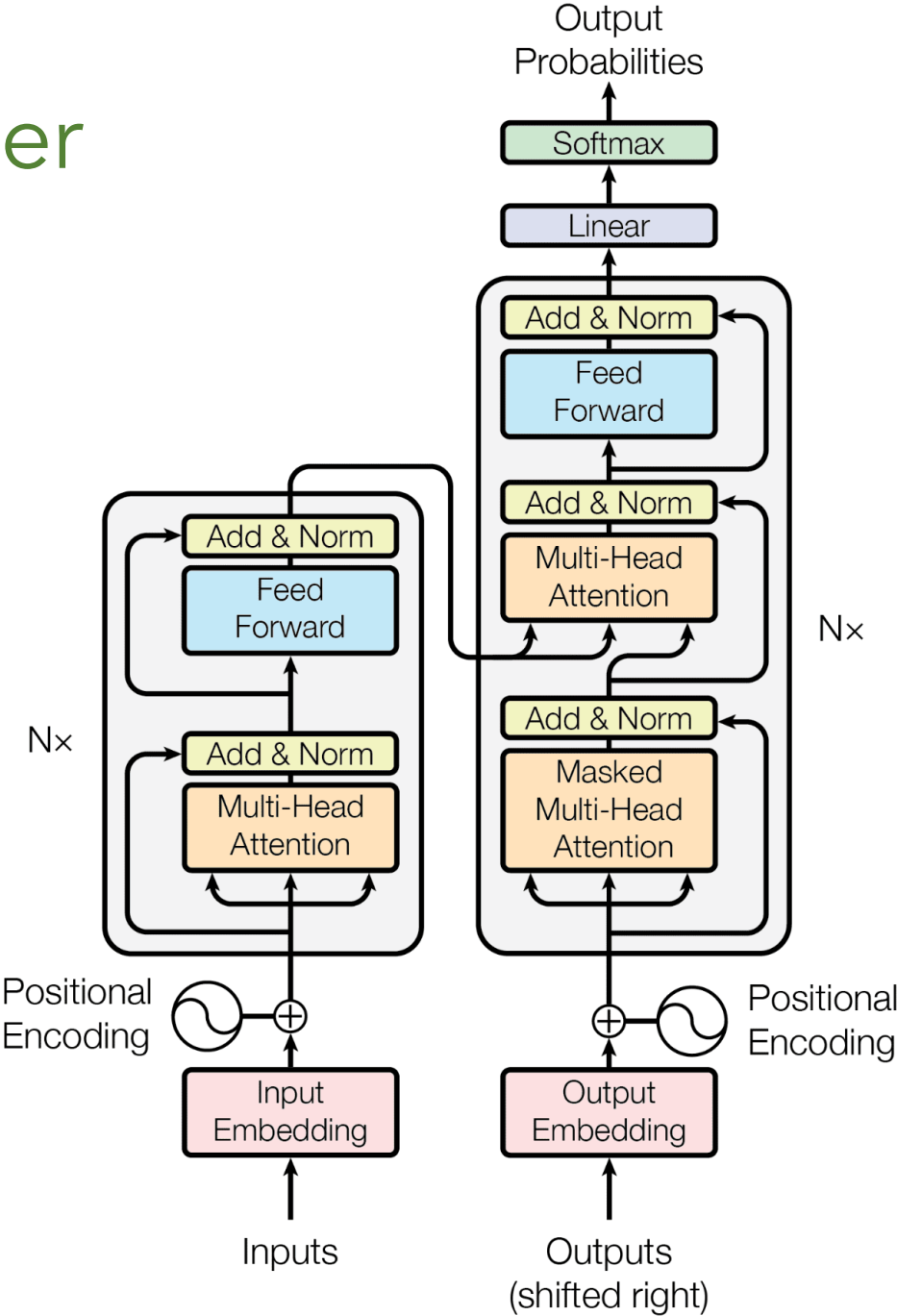
Lukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin*
illia.polosukhin@gmail.com

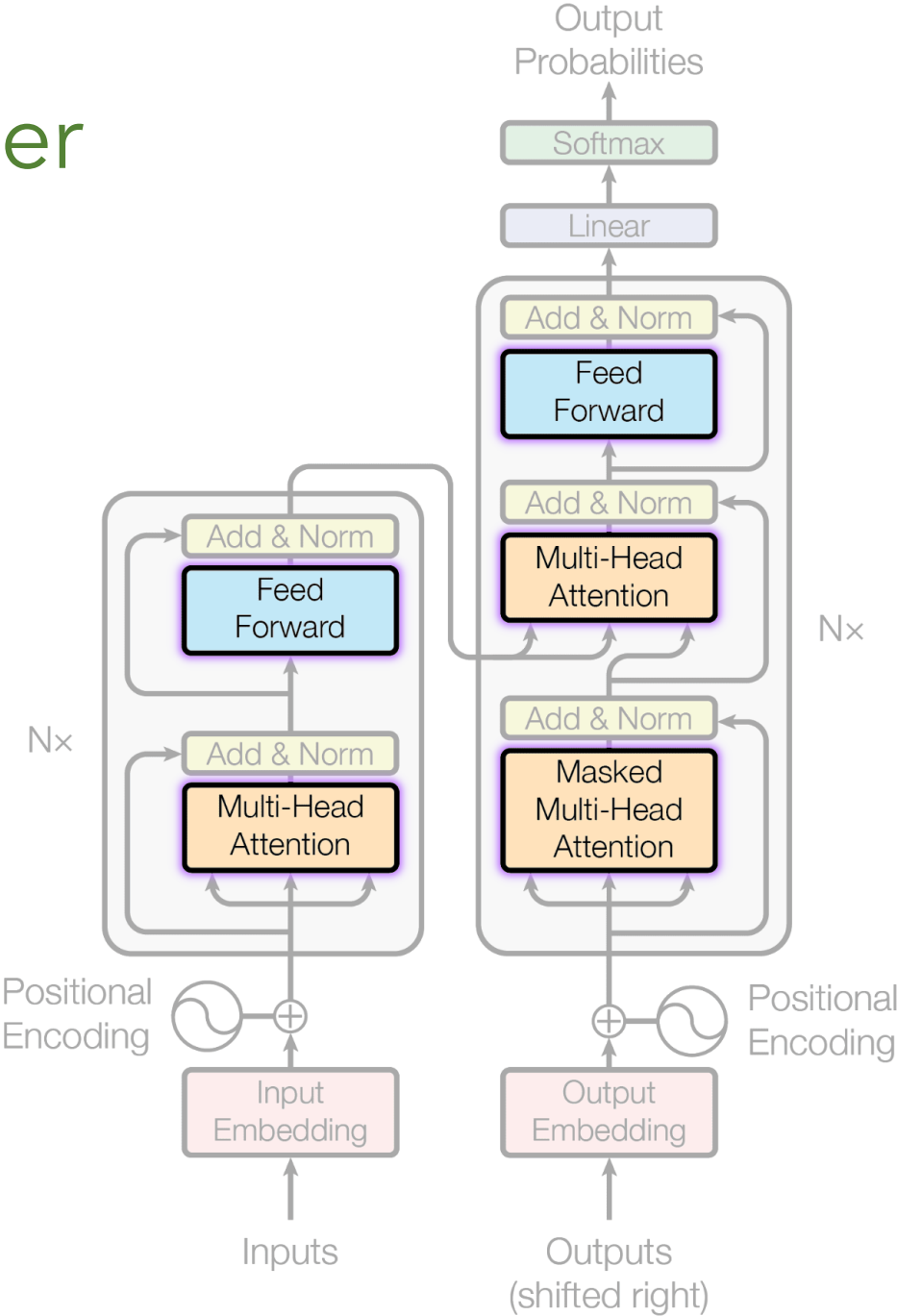
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention

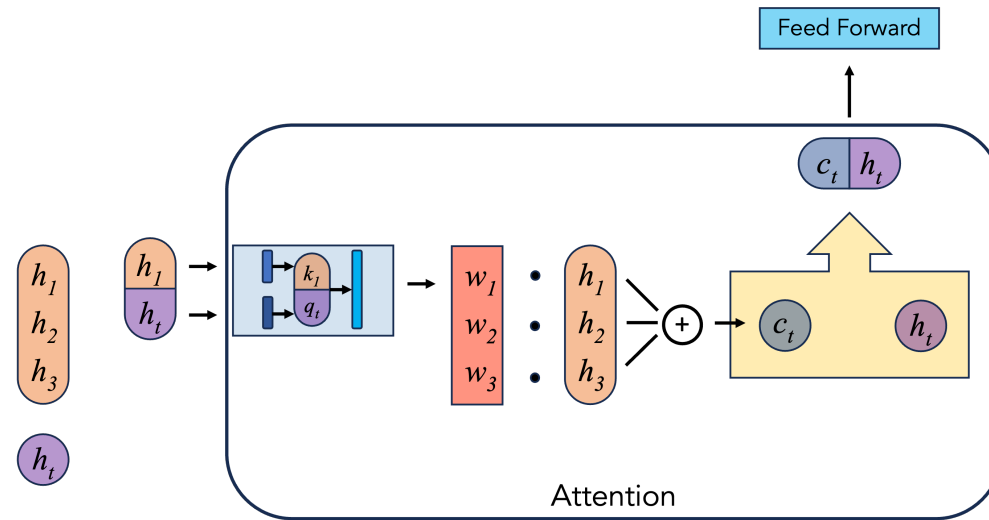
The Transformer



The Transformer



The Evolution of Attention



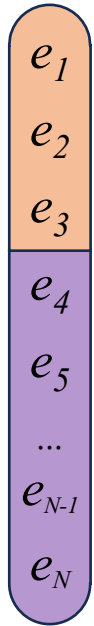
Hidden states generated by RNNs

Alignment NN layers convert:

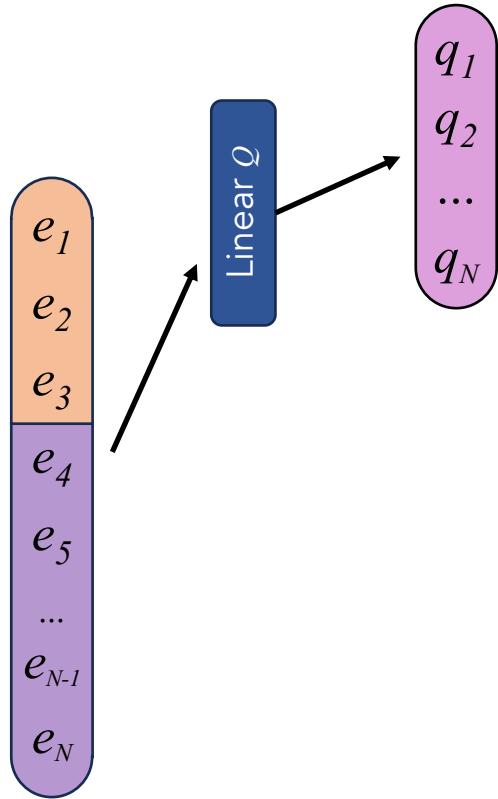
- encoder hidden states to *keys*
- decoder hidden states to *queries*
- *keys & queries* to *weights*

weights used to average unmodified encoder hidden states

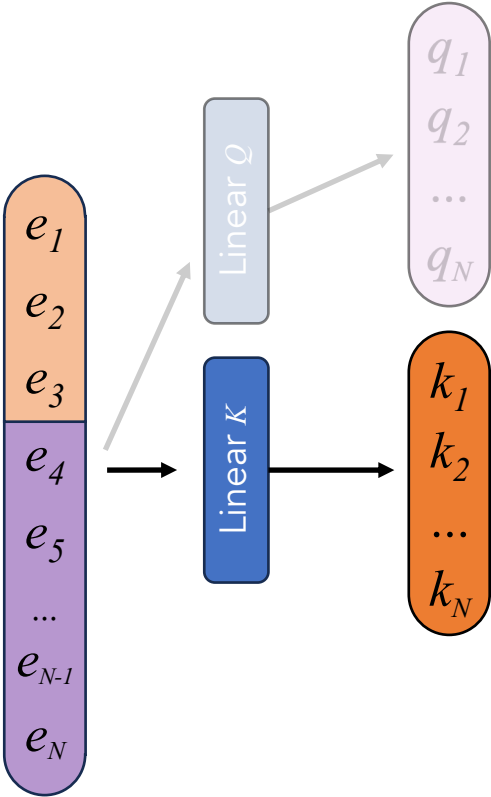
The Evolution of Attention



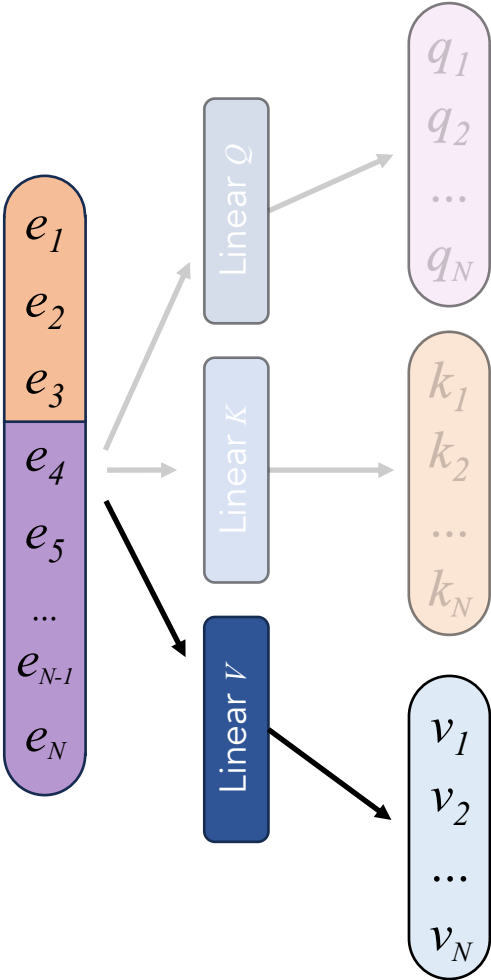
The Evolution of Attention



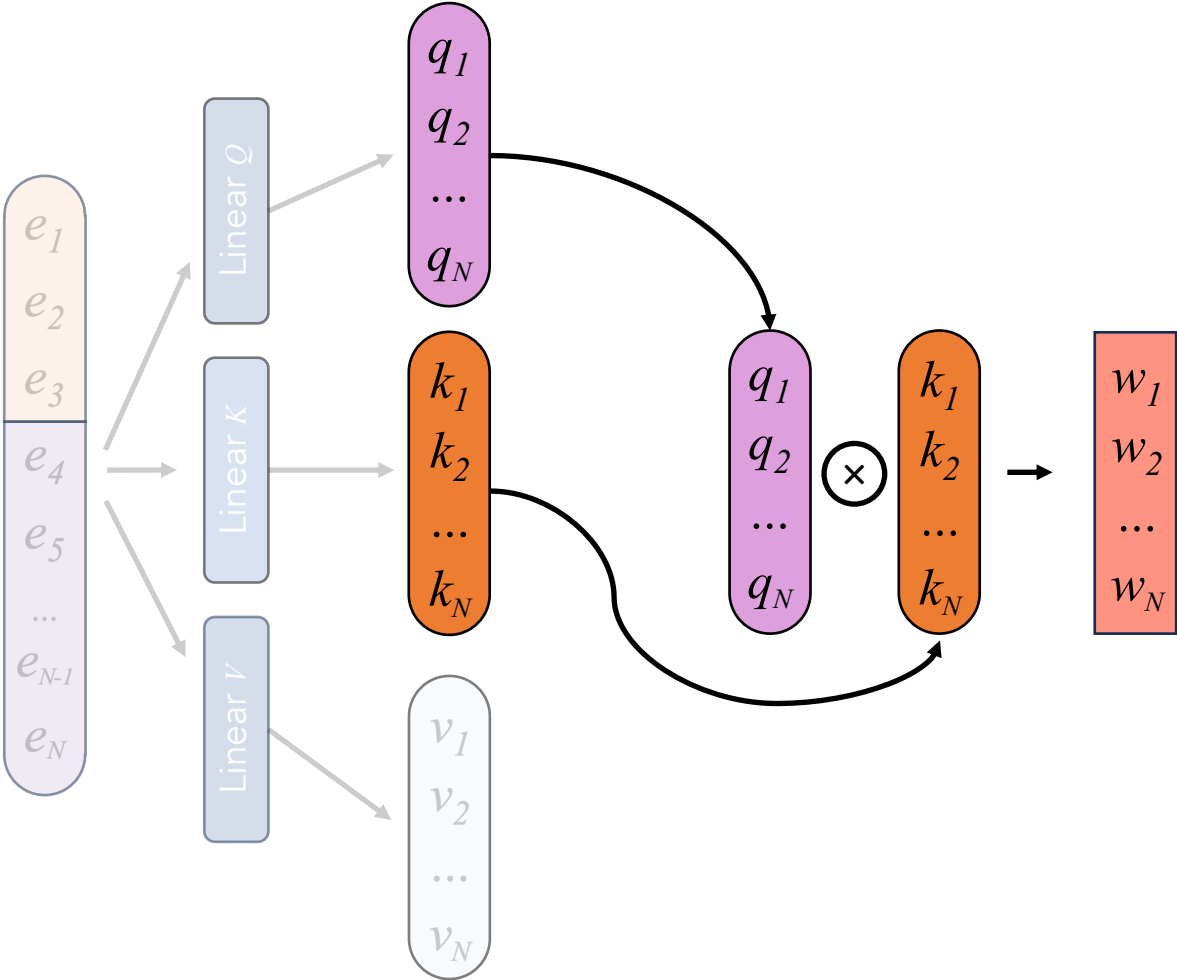
The Evolution of Attention



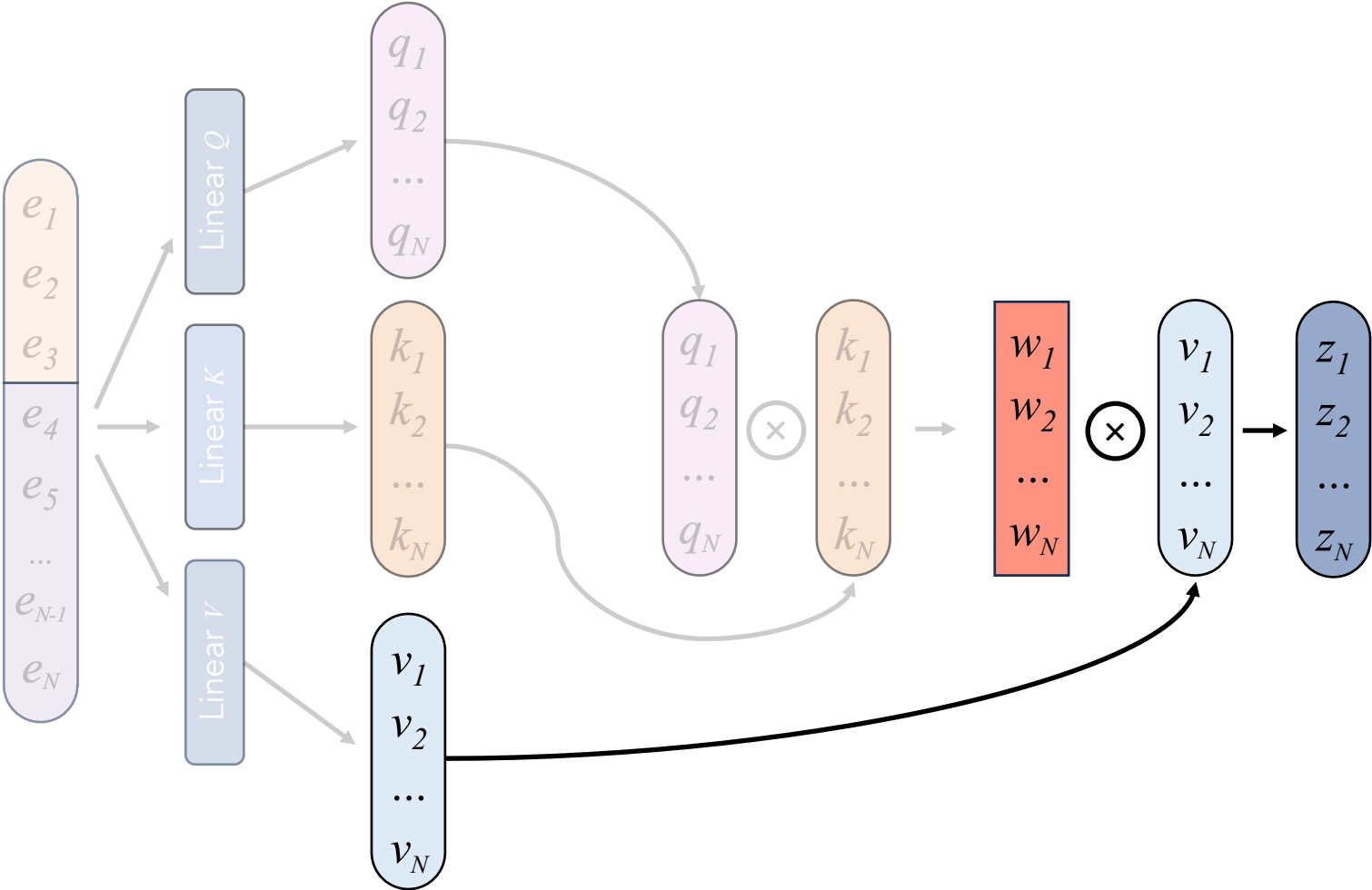
The Evolution of Attention



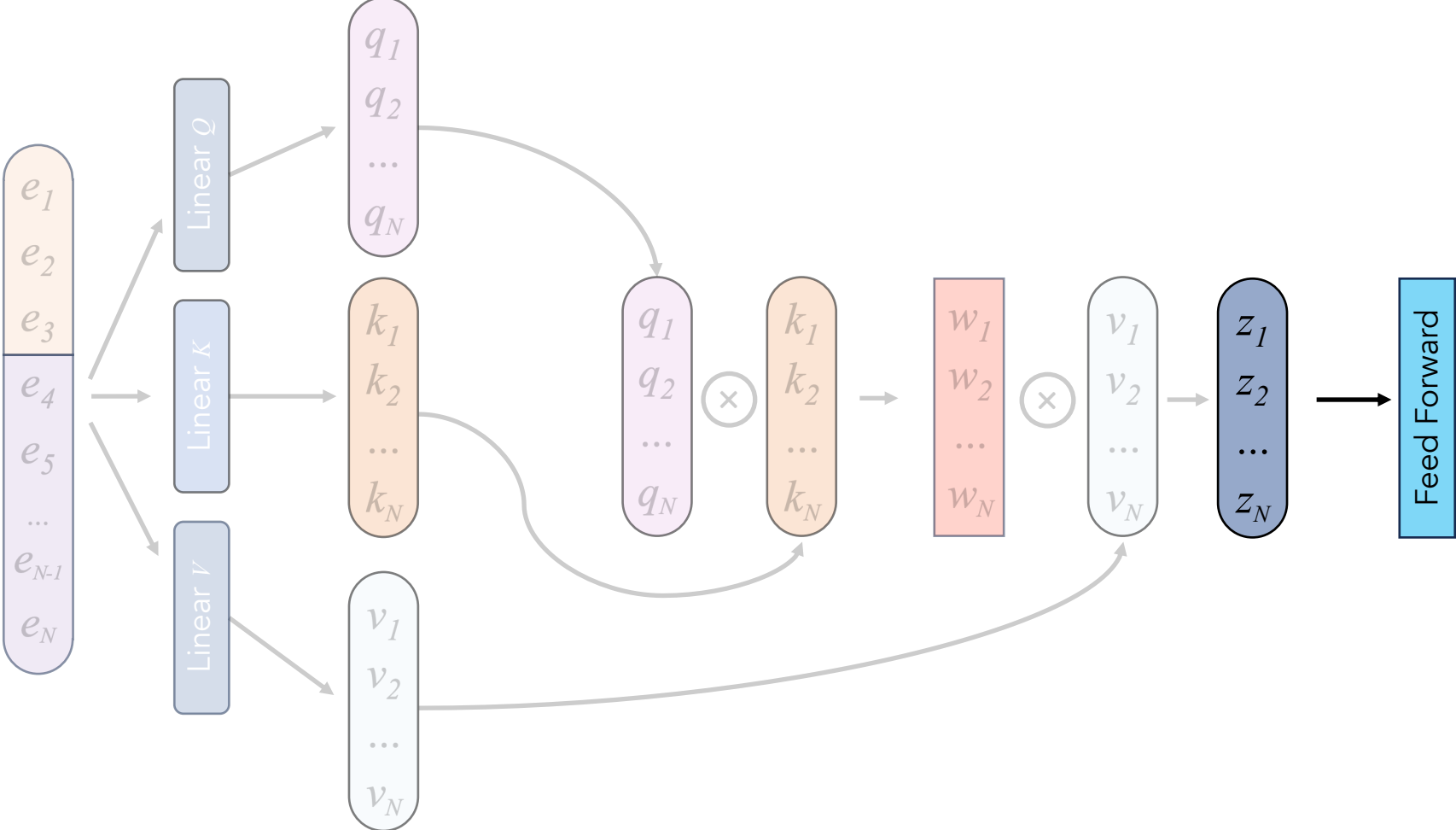
The Evolution of Attention



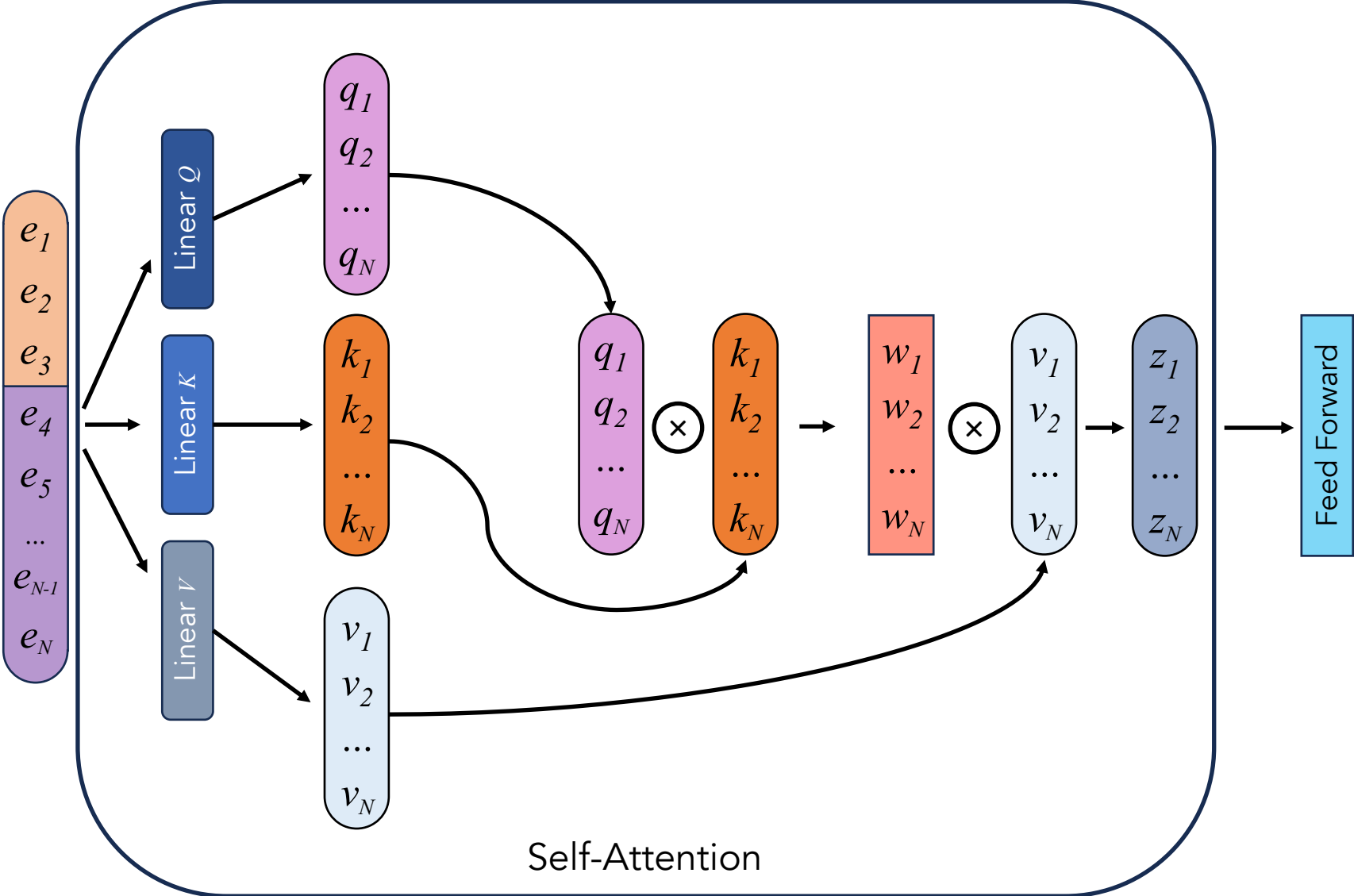
The Evolution of Attention



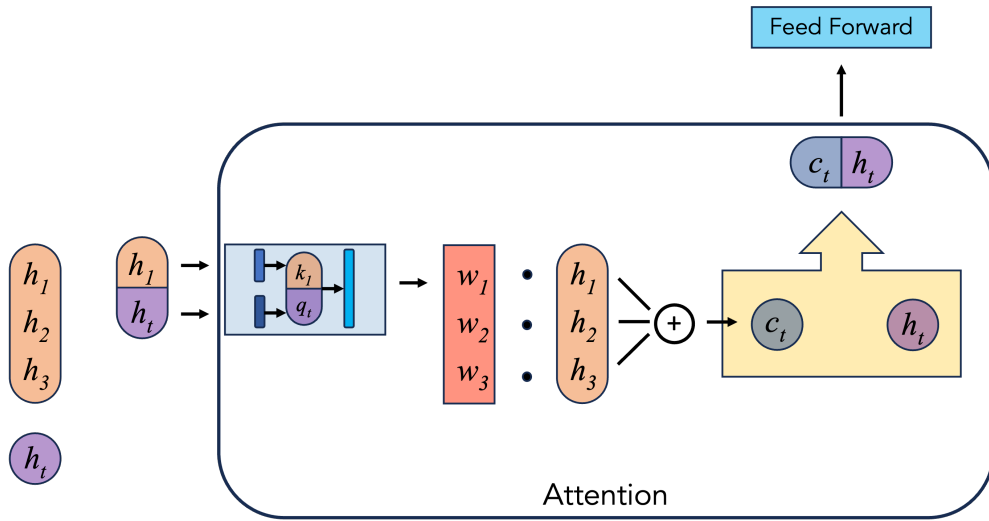
The Evolution of Attention



The Evolution of Attention



The Evolution of Attention

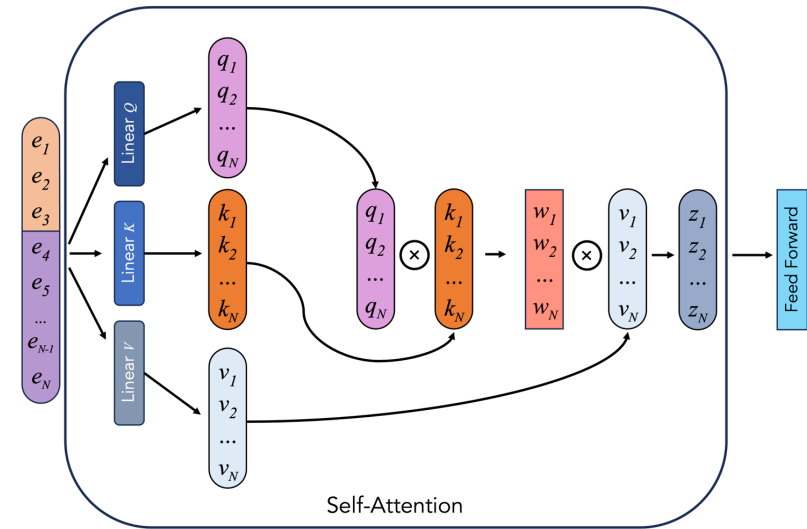


Hidden states generated by RNNs

Alignment NN layers convert:

- encoder hidden states to *keys*
- decoder hidden states to *queries*
- *keys & queries* to *weights*

weights used to average unmodified encoder hidden states

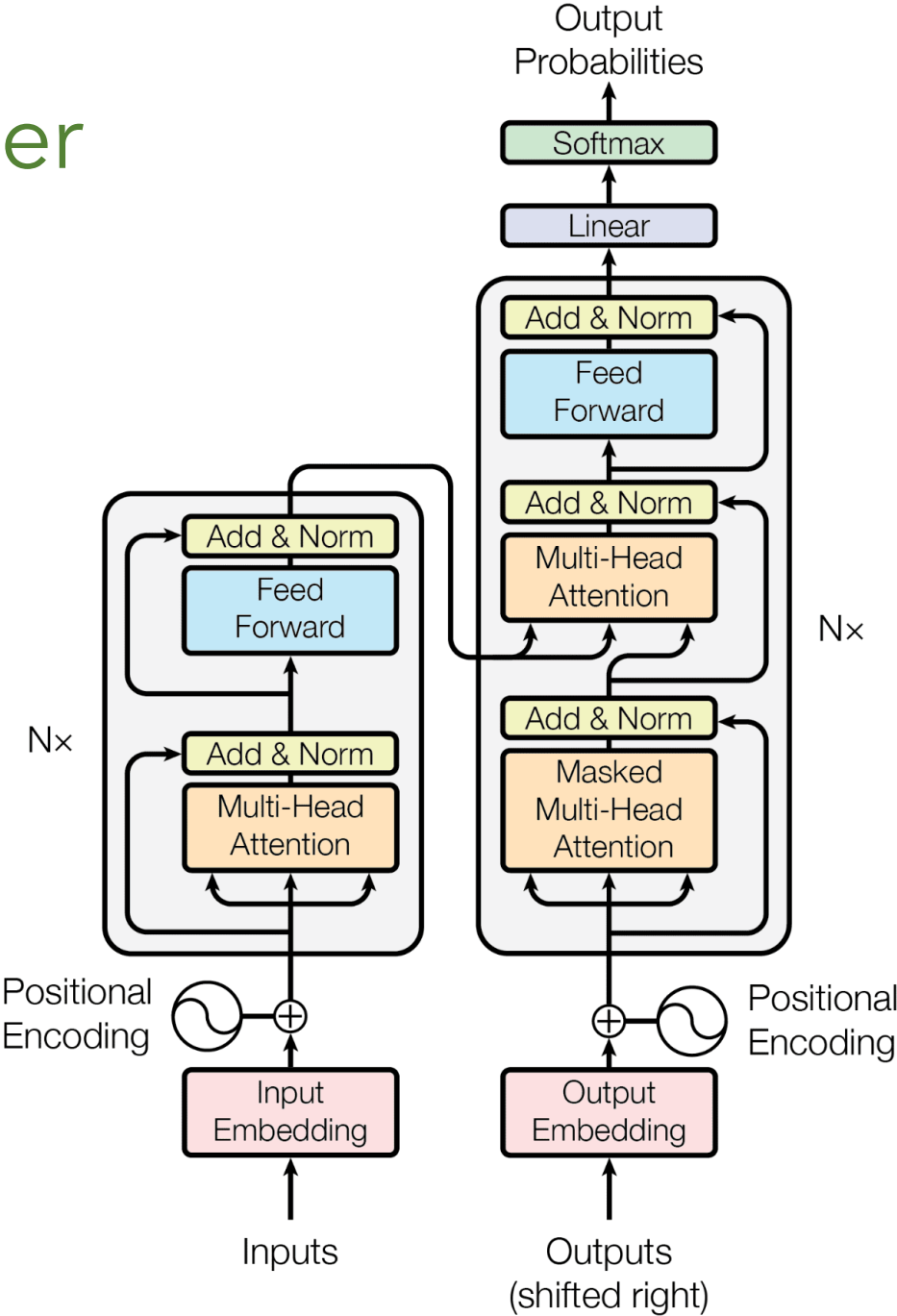


Linear NNs directly create *keys, queries, and values*

Matrix mult converts *keys & queries* to *weights*

weights used to average *values*

The Transformer



Multi-Head Attention

