Foundations of Deep Learning HW2

Yonatan Ariel Slutzky Eyal Grinberg

1 May 2023

1 Part 1

1.1 Question 1

We'll show that for $\forall f \in \mathcal{Y}^{\mathcal{X}}$, the width B required to realize it using the shallow network isn't greater than 2^{d-1}.
We'll denote the following:

$$\forall i \in \{-1,1\}.f^{-1}[i] := \{x \in \{-1,1\}^d | f(x) = i\}$$

In class we saw that the XOR function requires 2^{d-1} AND gates to be expressed by the shallow network.
One weight assignment that produces the XOR function is the following;

$$\forall \tilde{x} \in XOR^{-1}[1].W_{\tilde{x}} = \tilde{x}$$

In other words, each AND gate is assigned to an input from $XOR^{-1}[1]$, its weights will be the input itself, and the output from said gate is

$$AND(\langle W_{\tilde{x}}, x\rangle) = 2\chi_{x=\tilde{x}} - 1$$

Thus, with said weight assignment, the shallow network does produce XOR.
Given $f \in \mathcal{Y}^{\mathcal{X}}$, we'll want to modify the representation above of XOR in such a manner that it will express f, without adding any gates. This will prove that the shallow network can express $\mathcal{Y}^{\mathcal{X}}$ with a width of no more than 2^{d-1}.
We'll first want to address the inputs in $f^{-1}[1] \setminus XOR^{-1}[1]$. These inputs receive an output of -1 in XOR, but need to be receive an output of 1 in f.
We'll denote the following mapping $g : f^{-1}[1] \setminus XOR^{-1}[1] \to XOR^{-1}[1]$:

$$g(\tilde{x}) = \tilde{x} \text{ with its first bit flipped}$$

It's obvious that $Im(g) \subseteq XOR^{-1}[1]$ since $f^{-1}[1] \setminus XOR^{-1}[1] \subseteq XOR^{-1}[-1]$.
We'll also note that g is one-to-one.
Lastly, we'll note that $d(\tilde{x}, g(\tilde{x})) = 1$ where $d(.,.)$ is the distance in bits.
The first change to the weight assignment above, will be to "include" each $\tilde{x} \in f^{-1}[1] \setminus XOR^{-1}[1]$ in $g(\tilde{x})$'s gate - this will be done by changing $g(\tilde{x})$'s weight vector such that it includes a "don't care" in the difference bit (=the

first bit) between $\tilde{x}$ and $g(\tilde{x})$. We'll denote the result weights $\tilde{W}$.
In this setting, the output of the AND gate of $g(\tilde{x})$ will now be

$$AND(\langle \tilde{W}_{g(\tilde{x})}, x \rangle) = 2\chi_{x=g(\tilde{x}) \vee x=\tilde{x}} - 1$$

After this modification, all values of $f^{-1}[1]$ receive the output of 1 in the network.
We are left to address the inputs in $f^{-1}[-1] \setminus XOR^{-1}[-1]$.
For each $\tilde{x} \in f^{-1}[-1] \setminus XOR^{-1}[-1]$, we'll check if the weight vector of $\tilde{x}$'s AND gate (which exists since $\tilde{x} \in XOR^{-1}[1]$) includes a "don't care". If not, we'll simply remove the gate from the network. Otherwise, we know another input was "included" in the gate, and thus we'll set the weight vector to be that input. In any case, $\tilde{x}$ won't "turn on" any of the remaining gates, and thus will receive the output of -1.
In total, no gates were added in the process, and all inputs now receive their appropriate outputs according to f as desired. $\square$
p.s. - in the extreme case where $f \equiv -1$, we'll represent f using a single AND gate with a "don't care" weight vector (such that all inputs don't "turn on" the gate).

1.2 Question 2

We'll first note that

$$|\mathcal{Y}^{\mathcal{X}}| = |\mathcal{Y}|^{|\mathcal{X}|} = 2^{2^d}$$

Therefore, a necessary condition for $\overline{\mathcal{H}_{\overline{\mathcal{B}}}}$ to realize $\mathcal{Y}^{\mathcal{X}}$ is that

$$|\overline{\mathcal{H}_{\overline{\mathcal{B}}}}| \geq 2^{2^d}$$

This is because each function in $\mathcal{Y}^{\mathcal{X}}$ must be realized by at-least one weight assignment in $\overline{\mathcal{H}_{\overline{\mathcal{B}}}}$.
We'll devise a upper bound on $|\overline{\mathcal{H}_{\overline{\mathcal{B}}}}|$ that relies on $\overline{\mathcal{B}}$.
We'll show that if that upper bound is no less than 2^{2^d} then $\overline{\mathcal{B}} \in exp(d)$. This will prove that the condition $\overline{\mathcal{B}} \in exp(d)$ is required in order for $\overline{\mathcal{H}_{\overline{\mathcal{B}}}}$ to realize $\mathcal{Y}^{\mathcal{X}}$ (this isn't a sufficient condition).
WLOG we'll treat $\overline{\mathcal{B}}$ as no lesser than d.
For each AND gate in the deep network, there are at most $\overline{\mathcal{B}}$ weight entries which receive values from $\{-1, 0, 1\}$. Therefore, each AND gate has at most $3^{\overline{\mathcal{B}}}$ configurations.
Since the network is fully connected, the ordering of the different configurations in each layer doesn't affect the amount of functions represented by the network, in oppose to the identity of the different configurations chosen. Thus, since each AND layer "picks" at most $\overline{\mathcal{B}}$ weight configurations with no regards to order and with repetitions, the total amount of different possible combinations for each AND layer is no more than

$$\binom{\overline{\mathcal{B}} + 3^{\overline{\mathcal{B}}} - 1}{\overline{\mathcal{B}}}$$

The case is similar with the OR layers, only they "pick" weight entries from $\{0,1\}$. Thus, the total amount of different possible combinations for each OR layer is also no more than

$$\binom{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}{\overline{\mathcal{B}}}$$

Since each layer is "independent" of the others, and since there are a total of $2log(d)$ layers, the upper bound for $|\overline{\mathcal{H}_{\overline{\mathcal{B}}}}|$ is

$$\binom{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}{\overline{\mathcal{B}}}^{2log(d)} = (\frac{(\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1)!}{(\overline{\mathcal{B}})!(3^{\overline{\mathcal{B}}}-1)!})^{2log(d)} = (*)$$

Stirling's bound states that $\forall n \in \mathbb{N}.n! < \sqrt{2\pi n}(\frac{n}{e})^n e^{\frac{1}{12n}} < \sqrt{8\pi n}(\frac{n}{e})^n$. Plugging it into $(*)$ results in

$$(*) < (\frac{\sqrt{8\pi(\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1)}(\frac{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}{e})^{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}}{\sqrt{8\pi\overline{\mathcal{B}}}(\frac{\overline{\mathcal{B}}}{e})^{\overline{\mathcal{B}}}\sqrt{8\pi(3^{\overline{\mathcal{B}}}-1)}(\frac{3^{\overline{\mathcal{B}}}-1}{e})^{(3^{\overline{\mathcal{B}}}-1)}})^{2logd} =$$

$$= (\sqrt{\frac{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}{8\pi\overline{\mathcal{B}}(3^{\overline{\mathcal{B}}}-1)}}\frac{(\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1)^{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}}{\overline{\mathcal{B}}^{\overline{\mathcal{B}}}(3^{\overline{\mathcal{B}}}-1)^{3^{\overline{\mathcal{B}}}-1}})^{2log(d)} = (**)$$

It holds that $\forall n \in \mathbb{N}.\sqrt{\frac{n+3^n-1}{8\pi n(3^n-1)}} < 1$ and thus

$$(**) < (\frac{(\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1)^{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}}{\overline{\mathcal{B}}^{\overline{\mathcal{B}}}(3^{\overline{\mathcal{B}}}-1)^{3^{\overline{\mathcal{B}}}-1}})^{2log(d)} = (***)$$

Newton's binomial states that $\forall n \in \mathbb{N}, 0 < x < y \in \mathbb{R}.(x+y)^n \leq (n+1)y^n$ and thus since $0 < \overline{\mathcal{B}} < 3^{\overline{\mathcal{B}}} - 1$ we get that

$$(***) \leq (\frac{(\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}})(3^{\overline{\mathcal{B}}}-1)^{\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}}-1}}{\overline{\mathcal{B}}^{\overline{\mathcal{B}}}(3^{\overline{\mathcal{B}}}-1)^{3^{\overline{\mathcal{B}}}-1}})^{2log(d)} = (\frac{(\overline{\mathcal{B}}+3^{\overline{\mathcal{B}}})(3^{\overline{\mathcal{B}}}-1)^{\overline{\mathcal{B}}}}{\overline{\mathcal{B}}^{\overline{\mathcal{B}}}})^{2log(d)} \leq$$

$$\leq (3^{\overline{\mathcal{B}}^2}3^{\overline{\mathcal{B}}^2})^{2log(d)} = 3^{4\overline{\mathcal{B}}^2 log(d)} \leq 2^{8\overline{\mathcal{B}}^2 log(d)}$$

We'll want to find the values for $\overline{\mathcal{B}}$ for which the last term is at-least 2^{2^d}:

$$2^{2^d} \leq 2^{8\overline{\mathcal{B}}^2 log(d)} \rightarrow 2^d \leq 8\overline{\mathcal{B}}^2 log(d) \rightarrow \frac{2^{d-3}}{log(d)} \leq \overline{\mathcal{B}}^2 \rightarrow \frac{2^{\frac{d-3}{2}}}{\sqrt{log(d)}} \leq \overline{\mathcal{B}}$$

To summarize, we've shown the following logical implications:
If the deep network realizes all of $\mathcal{Y}^{\mathcal{X}}$ then $|\overline{\mathcal{H}_{\overline{\mathcal{B}}}}| \geq 2^{2^d}$
If $|\overline{\mathcal{H}_{\overline{\mathcal{B}}}}| \geq 2^{2^d}$ then $2^{8\overline{\mathcal{B}}^2 log(d)} \geq 2^{2^d}$
If $2^{8\overline{\mathcal{B}}^2 log(d)} \geq 2^{2^d}$ then $\overline{\mathcal{B}} = exp(d)$ as desired. $\square$

2 Part 2

2.1 Question 1

Let $g : \mathbb{R} \to \mathbb{R}$ be a piece-wise linear function with $k \in [\mathcal{B}] \cup \{0\}$ pieces. We'll prove g can be realized by $\mathcal{H}_{\mathcal{B}}$.
g is piece-wise linear, and thus there exist $-\infty = z_0 < z_1 < .. < z_{k-1} < z_k = \infty$ such that for $\forall m \in [k]$ there exist $\alpha_m, \beta_m \in \mathbb{R}$ such that

$$\forall x \in (z_{m-1}, z_m).g(x) = \alpha_m x + \beta_m$$

Since g is continuous, it holds that

$$\forall m \in [k-1].\alpha_m z_m + \beta_m = g(z_m) = \alpha_{m+1} z_m + \beta_{m+1} \to z_m = \frac{\beta_{m+1} - \beta_m}{\alpha_m - \alpha_{m+1}} \ (*)$$

We'll show that we can find a weight assignment such that in each interval, the output of the network for each input in said interval is the same as the output of g. The following are the selected weights:

$$w_1^{(1)} := -1, w_1^{(2)} := \alpha_1, b_1^{(1)} := z_1, b^{(2)} := \beta_1 + \alpha_1 z_1$$

$$w_2^{(1)} := 1, w_2^{(2)} := \alpha_2, b_2^{(1)} := \frac{\beta_2 - \beta_1 - \alpha_1 z_1}{\alpha_2}$$

$$\forall m \in [k-1] \setminus [1].w_{m+1}^{(1)} := 1, w_{m+1}^{(2)} := \alpha_{m+1} - \alpha_m, b_{m+1}^{(1)} := \frac{\beta_{m+1} - \beta_m}{\alpha_{m+1} - \alpha_m}$$

$$\forall m \in [\mathcal{B}] \setminus [k].w_m^{(1)} := 0, w_m^{(2)} := 0, b_m^{(1)} := 0$$

First, it holds from $(*)$ that

$$b_2^{(1)} = \frac{\beta_2 - \beta_1 - \alpha_1 z_1}{\alpha_2} = \frac{\beta_2 - \beta_1 - \alpha_1(\frac{\beta_2 - \beta_1}{\alpha_1 - \alpha_2})}{\alpha_2} = \frac{(\beta_2 - \beta_1)(1 - \frac{\alpha_1}{\alpha_1 - \alpha_2})}{\alpha_2} =$$

$$= \frac{(\beta_2 - \beta_1)(\frac{\alpha_1 - \alpha_2 - \alpha_1}{\alpha_1 - \alpha_2})}{\alpha_2} = \frac{(\beta_2 - \beta_1)(\frac{-\alpha_2}{\alpha_2})}{\alpha_1 - \alpha_2} = -z_1$$

We also get from $(*)$ that for $\forall m \in [k-1] \setminus [1]$

$$b_{m+1}^{(1)} = -z_m$$

Thus, the following holds:
1. For $\forall x \in (z_0, z_1) = (-\infty, z_1)$

$$w_1^{(1)} x + b_1^{(1)} = -x + z_1 > 0$$

$$\forall x \in (z_0, z_1)^{\mathcal{C}} = [z_1, \infty).w_1^{(1)} x + b_1^{(1)} = -x + z_1 \leq 0$$

2. For $\forall m \in [k-1], x \in (z_m, \infty)$.

$$w_{m+1}^{(1)} x + b_{m+1}^{(1)} = x - z_m > 0$$

$$\forall x \in (z_m, \infty)^{\mathcal{C}} = (-\infty, z_m].w_{m+1}^{(1)}x + b_{m+1}^{(1)} = x - z_m \leq 0$$

Thus, the proposed construction ensures that the first set of weights is "active" only in the first interval and is the only set "active" there (active in the sense of the $ReLU$), and for every interval afterwards, we begin with the second set being "active", and we keep adding "active" weights each interval.
We'll show using induction on the interval's index that the output of the network is indeed the same as g's.
$\underline{m = 1:}$ The output of the network for $\forall x \in (z_0, z_1)$ is

$$w_1^{(2)}w_1^{(1)}x + w_1^{(2)}b_1^{(1)} + b^{(2)} = -\alpha_1(-1)x - \alpha_1 z_1 + \beta_1 + \alpha_1 z_1 = \alpha_1 x + \beta_1 = g(x)$$

$\underline{m = 2:}$ The output of the network for $\forall x \in (z_1, z_2)$ is

$$w_2^{(2)}w_2^{(1)}x + w_2^{(2)}b_2^{(1)} + b^{(2)} = \alpha_2 x - \alpha_2 z_1 + \beta_1 + \alpha_1 z_1 = \alpha_2 x + z_1(\alpha_1 - \alpha_2) + \beta_1 =$$

$$\alpha_2 x + \beta_2 - \beta_1 + \beta_1 = \alpha_2 x + \beta_2 = g(x)$$

$\underline{m \to m+1, m+1 \in [k]}$: Suppose the argument is true for m. Thus, we have that all the weight sets from the second until the mth are active in the $m+1'$th interval. We also have from the inductive assumption that

$$\sum_{j=2}^{m} w_j^{(2)}w_j^{(1)} = \alpha_m$$

$$\sum_{j=2}^{m} w_j^{(2)}b_j^{(1)} + b^{(2)} = \beta_m$$

Thus, the output of the network for $\forall x \in (z_m, z_{m+1})$ is

$$\sum_{j=2}^{m+1}(w_j^{(2)}w_j^{(1)}x + w_j^{(2)}b_j^{(1)}) + b^{(2)} = \sum_{j=2}^{m}(w_j^{(2)}w_j^{(1)}x + w_j^{(2)}b_j^{(1)}) + b^{(2)} + w_{m+1}^{(2)}w_{m+1}^{(1)}x + w_{m+1}^{(2)}b_{m+1}^{(1)} = (**)$$

From the construction described above we get that

$$(**) = \alpha_m x + (\alpha_{m+1} - \alpha_m)x + \beta_m + (\alpha_{m+1} - \alpha_m)\frac{\beta_{m+1} - \beta_m}{\alpha_{m+1} - \alpha_m} = \alpha_{m+1}x + \beta_{m+1} = g(x)$$

Applying the induction principal completes the section.
Thus, we got that if g has no more than $\mathcal{B}$ pieces it is realizable by the shallow network.
We are left to prove that any mapping f realized by the shallow network is piece-wise linear with no more than $\mathcal{B} + 1$ pieces.
We'll denote the weights used to realize f:

$$w^{(2)}, w^{(1)}, b^{(1)} \in \mathbb{R}^{\mathcal{B}}, b^{(2)} \in \mathbb{R}$$

We'll explicitly write f:

$$\forall x \in \mathbb{R}. f(x) = \sum_{j=1}^{\mathcal{B}} w_j^{(2)} max(0, w_j^{(1)} x + b_j^{(1)}) + b^{(2)} = (***)$$

f is a linear combination of continuous functions, and thus it is a continuous function. Next, we get that

$$(***) = \sum_{j \in \{j \in [\mathcal{B}] | w_j^{(1)} x + b_j^{(1)} > 0\}} w_j^{(2)} w_j^{(1)} x + w_j^{(2)} b_j^{(1)} + b^{(2)}$$

For every $x \in \mathbb{R}$, f is a linear combination of linear functions $f_j(x) := w_j^{(1)} x + b_j^{(1)}$ for $j \in [\mathcal{B}]$.
If f_j is decreasing, then it will be "active" in the sense of $ReLU$ up until its intersection with the x axis, from which point it will be "inactive".
If f_j is increasing, then it will be "inactive" in the sense of $ReLU$ up until its intersection with the x axis, from which point it will be "active".
If f_j is constant, it will be "active" in sense of $ReLU$ if and only if it's positive.
Thus, each f_j can only transfer at most once, from "active" to "inactive" or from "inactive" to "active".
Therefore, f takes the form of at most $\mathcal{B}+1$ different linear combinations (since there are at most $\mathcal{B}$ "changes"). Combining with the fact that f is continuous completes this section. $\square$
The condition for $B \geq 2$ is required because otherwise, the output is of the form

$$w^{(2)} max\{0, w^{(1)} x + b^{(1)}\} + b^{(2)} = \begin{cases} max\{0, w^{(2)} w^{(1)} x + w^{(2)} b^{(2)}\} & w^{(2)} \geq 0 \\ min\{0, w^{(2)} w^{(1)} x + w^{(2)} b^{(2)}\} & w^{(2)} < 0 \end{cases}$$

The result output will always be 2-piece linear with one of the pieces being a constant 0, and thus the network won't realize any 1-piece non-constant linear function.

2.2 Question 2

Suppose the linear mapping f has k linear pieces.
Since each piece is linear, if a piece "catches" 2 consecutive intervals, any subsequent interval the piece "catches" is of the type of the second interval it caught. Equivalently, every interval, reached by the piece, which is from the type of the first interval caught, is missed by said piece.
For instance, if the piece first caught an interval from $S_>$, then an interval from $S_<$, and then another interval, the third interval must be from $S_<$, and the interval right after the second one caught was missed.
Therefore, for every triplet of continuous intervals caught by f, f has 2 linear pieces on said triplet, where the middle interval was caught using both pieces.
The result is that the maximal amount of intervals f catches, is no more than the entirety of either $S_<$ or $S_>$, plus the amount of intervals from the other set

caught by the k pieces - the latter is no more than $\lceil \frac{k}{2} \rceil$.
Since $|S_>| = 2^{L-2}, |S_<| = 2^{L-2} + 1$, we get that the total amount of intervals caught is no more than

$$2^{L-2} + 1 + \lceil \frac{k}{2} \rceil \leq \lceil \frac{1}{2}(2^{L-1} + 1) + \frac{1}{2}(k+1) \rceil$$

Proving the argument. $\square$

2.3 Question 3

We have shown in class that a Leaky-ReLU neuron can be expressed as a linear combination of two ReLU neurons, and that a ReLU neuron can be expressed as a linear combination of two Leaky-ReLU neurons.
Namely, we have shown that there exist $\alpha, \beta, \alpha', \beta' \in \mathbb{R}$ such that

$$ReLU(x) \equiv \alpha LeakyReLU_a(x) + \beta LeakyReLU_a(x)$$

$$LeakyReLU_a(x) \equiv \alpha' ReLU(x) + \beta' ReLU(x)$$

We'll denote $\mathcal{H}^*$ to be the classes of fully connected NNs with Leaky-ReLU activation. For instance, $\mathcal{H}^*_{\mathcal{B}}$ is the class of shallow leaky ReLU fully connected NNs with width $\mathcal{B}$.
We'll show that for any $h \in \mathcal{H}_{\mathcal{B}}$, h can be realized by $\mathcal{H}^*_{2\mathcal{B}}$. An exact proof exists for the opposite case, and proofs in the same nature (with some more technical aspects) exist for the deep cases.
We'll denote $h(x) := w^{(2)} ReLU(w^{(1)} x + b^{(1)}) + b^{(2)}$ $(*)$ with element-wise ReLU and with

$$w^{(2)} \in \mathbb{R}^{1 \times \mathcal{B}}, w^{(1)} \in \mathbb{R}^{\mathcal{B} \times 1}, b^{(1)} \in \mathbb{R}^{\mathcal{B} \times 1}, b^{(2)} \in \mathbb{R}$$

Opening the product, we have that

$$(*) = \sum_{i=1}^{\mathcal{B}} w_i^{(2)} ReLU(w_i^{(1)} x + b_i^{(1)}) + b^{(2)} =$$

$$= \sum_{i=1}^{\mathcal{B}} w_i^{(2)} (\alpha LeakyReLU_a(w_i^{(1)} x + b_i^{(1)}) + \beta LeakyReLU_a(w_i^{(1)} x + b_i^{(1)})) + b^{(2)} = (**)$$

We'll denote the following weights for a realization $\hat{h} \in \mathcal{H}^*_{2\mathcal{B}}$:

$$\forall j \in [2\mathcal{B}]. \hat{w}_j^{(1)} := w^{(1)}_{\lfloor \frac{j+1}{2} \rfloor}, \hat{b}_j^{(1)} := b^{(1)}_{\lfloor \frac{j+1}{2} \rfloor}$$

$$\forall j \in [2\mathcal{B}]. \hat{w}_j^{(2)} := w^{(2)}_{\lfloor \frac{j+1}{2} \rfloor} (\alpha \chi_{j \in \mathbb{N}_{even}} + \beta \chi_{j \in \mathbb{N}_{odd}})$$

$$\hat{b}^{(2)} := b^{(2)}$$

Thus, we get that

$$(**) = \sum_{j=1}^{2\mathcal{B}} \hat{w}_j^{(2)} LeakyReLU_a(\hat{w}_j^{(1)} x + b_j^{(1)}) + \hat{b}^{(2)}$$

Therefore, $h(x) \equiv \hat{h}(x)$ as desired.
Hence, the following holds

$$\mathcal{H}_{\mathcal{B}} \subseteq \mathcal{H}^*_{2\mathcal{B}}, \overline{\mathcal{H}_{\overline{\mathcal{B}}}} \subseteq \overline{\mathcal{H}^*_{2\overline{\mathcal{B}}}}$$

$$\mathcal{H}^*_{\mathcal{B}} \subseteq \mathcal{H}_{2\mathcal{B}}, \overline{\mathcal{H}^*_{\overline{\mathcal{B}}}} \subseteq \overline{\mathcal{H}_{2\overline{\mathcal{B}}}}$$

Moreover, by using passthrough layers, we'll also get that

$$\mathcal{H}^*_{\mathcal{B}} \subseteq \overline{\mathcal{H}^*_{\overline{\mathcal{B}}}}$$

We saw in class that given a continuous $f : \mathbb{R} \to \mathbb{R}$ and $\epsilon > 0$, there exists $\mathcal{B} \in \mathbb{N}$ such that there exists $h \in \mathcal{H}_{\mathcal{B}}$ that ϵ-approximates f. Therefore, since $h \in \mathcal{H}^*_{2\mathcal{B}}$, we get that $\mathcal{H}^*_{\mathcal{B}}$ is universal w.r.t to continuous real functions (and thus the same applies to $\overline{\mathcal{H}^*_{\overline{\mathcal{B}}}}$).
To show that $\overline{\mathcal{H}^*_{\overline{\mathcal{B}}}}$ is exponentially expressively efficient w.r.t $\mathcal{H}^*_{\mathcal{B}}$, we will use the same sawteeth function g introduced in class.
Since $g^{\circ L-1}$ can be realized by $\overline{\mathcal{H}_3}$ with depth L, it can be realized by $\overline{\mathcal{H}^*_6}$ with depth L.
Assume on the contrary that it can be realized by $\mathcal{H}^*_{\mathcal{B}}$ with sub-exponential $\mathcal{B}$ w.r.t L. Then, it can also be realized by $\mathcal{H}_{2\mathcal{B}}$ - where $2\mathcal{B}$ is also sub-exponential w.r.t L, contradicting the proof seen in class.
Therefore, to realize $g^{\circ L-1}$ in $\mathcal{H}^*_{\mathcal{B}}$, $\mathcal{B}$ is required to be exponential in L. $\square$

2.4 Question 4

Let $f \in \mathcal{F}$ be a Riemann integrable function on $\mathcal{R}$. Thus, f is Riemann integrable on $[0, 1]$.
Therefore, f is bound in $[0, 1]$ and has a countable amount of discontinuity points in $[0, 1]$, all of which are of type "jump".
We'll denote $x_0 = 0$, and also denote the series of discontinuity points to be $\{x_i\}_{i=1}^{\infty}$ where $\forall i \in \mathbb{N} \cup \{0\}. x_i < x_{i+1}$ and $lim_{i\to\infty} x_i = 1$.
The last property is assumed WLOG as there must be a limit since all points reside in $[0, 1]$.
We'll assume WLOG that none of the discontinuity points are discontinuous from both sides. This is possible since the difference they produce doesn't effect $d(.,.)$. We can overcome this assumption by continuing the proof treating f' which is f with all of said points "flattened", and then since $d(f, f') = 0$ the results will still hold.
We'll also assume that all discontinuity points are discontinuous from the left and continuous from the right (this assumption is technical and the proof can be extended to overcome it).
Finally, we'll denote $f_{max} := max_{x\in[0,1]} f(x), f_{min} := min_{x\in[0,1]} f(x)$ which exist since f is bound. We'll assume $f_{max} > f_{min}$ otherwise the function is constant and thus it is obvious $\mathcal{H}_{\mathcal{B}}$ can realize it.
We'll show that $\mathcal{H}_{\mathcal{B}}$ can approximate f w.r.t $d(.,.)$, and thus from the expressive efficiency argument we'll get that $\overline{\mathcal{H}_{\overline{\mathcal{B}}}}$ can also realize f.

Let $\epsilon > 0$. We'll construct a continuous function g such that $d(f,g) < \frac{\epsilon}{2}$.
g is continuous and thus as we saw in class, there exists $\mathcal{B} \in \mathbb{N}$ such that there exists a function $h \in \mathcal{H}_{\mathcal{B}}$ such that $d(g,h) < \frac{\epsilon}{2}$. Applying the triangular inequality w.r.t to d will result in $d(f,h) < \epsilon$ as required.
We'll denote $c := \frac{\epsilon}{4(f_{max} - f_{min})} > 0$ and $m := \frac{\pi^2}{6} > 0$.
For $\forall i \in \mathbb{N} \cup \{0\}$, we'll denote $x_i^* := max\{x_{i+1} - \frac{c}{(i+1)^2 m}, \frac{x_i + x_{i+1}}{2}\} \in (x_i, x_{i+1})$.
We'll define g in the following manner - $\forall i \in \mathbb{N} \cup \{0\}, \forall x \in [x_i, x_{i+1}]$:

$$g(x) := \begin{cases} f(x) & x \in [x_i, x_i^*] \\ \frac{f(x_{i+1}) - f(x_i^*)}{x_{i+1} - x_i^*} x + f(x_{i+1}) - \frac{f(x_{i+1}) - f(x_i^*)}{x_{i+1} - x_i^*} x_{i+1} & x \in (x_i^*, x_{i+1}] \end{cases}$$

In other words, g follows f from x_i up until x_i^*, and from then it is a linear line to x_{i+1}. Thus, g is indeed continuous.
It holds that

$$d(f,g) = \int_0^1 |f(x) - g(x)| dx = \sum_{i=0}^{\infty} \int_{x_i}^{x_{i+1}} |f(x) - g(x)| dx =$$

$$= \sum_{i=0}^{\infty} \int_{x_i^*}^{x_{i+1}} |f(x) - g(x)| dx = (*)$$

The maximal distance between f and g cannot be more than $f_{max} - f_{min}$, since g always receives values in the image of f. Thus,

$$(*) \leq \sum_{i=0}^{\infty} |x_{i+1} - x_i^*| (f_{max} - f_{min}) \leq \sum_{i=0}^{\infty} \frac{c}{(i+1)^2 m} (f_{max} - f_{min}) =$$

$$= \frac{\frac{\epsilon}{4(f_{max} - f_{min})}}{\frac{\pi^2}{6}} (f_{max} - f_{min}) \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\epsilon}{4} \frac{1}{\frac{\pi^2}{6}} \frac{\pi^2}{6} = \frac{\epsilon}{4} < \frac{\epsilon}{2}$$

As mentioned above, this proves that there exists $\mathcal{B} \in \mathbb{N}$ such that there exists $h \in \mathcal{H}_{\mathcal{B}}$ such that $d(f,h) < \epsilon$. $\square$

3 Part 3

3.1 Question 1

In our proof, we will address $conv_{(l)}(j,\gamma)$ and $pool_{(l)}(\lfloor \frac{j}{2} \rfloor, \gamma)$ for $\forall l \in [L-1]$, $j \in [\frac{N}{2^l}], \gamma \in r_l$ as functions of the input $d_1, .., d_N$. It's important to note that not all inputs are necessarily included in these functions.
The tensor $\phi^{l,j,\gamma}$ will be the "tabularization" of $conv_{(l)}(j,\gamma)$, with each of its axis corresponding to an input variable that affects the function, and each entry represents the actual value of said input variable.
In other words, if the relevant input variables for $conv_{(l)}(j,\gamma)$ are indexed by $i_1, .., i_k \in [N]$, then the following holds

$$\phi^{l,j,\gamma}_{d_{i_1}, .., d_{i_k}} = conv_{(l)}(j,\gamma)|_{d_{i_1}, .., d_{i_k}}$$

Suppose the input is $d_1, .., d_N$.
We'll prove the desired argument with induction on the layer $l \in [L-1]$.
$\underline{l=1:}$ Let $j \in [\frac{N}{2}], \gamma \in [r_1]$. By definition, we have that

$$\phi^{1,j,\gamma}_{d_{2j-1},d_{2j}} = conv_{(1)}(j,\gamma)|_{d_1,..,d_N} = \langle a^{1,j,\gamma}, pool_{(0)}(j,:)\rangle|_{d_1,..,d_N} =$$

$$= \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha pool_{(0)}(j,\alpha)|_{d_1,..,d_N} = \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha conv_{(0)}(2j-1,\alpha)\cdot conv_{(0)}(2j,\alpha)|_{d_1,..,d_N} =$$

$$= \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha \langle a^{0,2j-1,\alpha}, rep(2j-1,:)\rangle \cdot \langle a^{0,2j,\alpha}, rep(2j,:)\rangle|_{d_1,..,d_N} =$$

$$= \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha (\sum_{d=1}^{M} a^{0,2j-1,\alpha}_d I_{d=d_{2j-1}}) \cdot (\sum_{d=1}^{M} a^{0,2j,\alpha}_d I_{d=d_{2j}})|_{d_1,..,d_N} =$$

$$= \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha a^{0,2j-1,\alpha}_{d_{2j-1}} \cdot a^{0,2j,\alpha}_{d_{2j}} = (\sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha a^{0,2j-1,\alpha} \otimes a^{0,2j,\alpha})_{d_{2j-1},d_{2j}}$$

Thus, $\phi^{1,j,\gamma} = \sum_{\alpha=1}^{r_0} a^{1,j,\gamma}_\alpha a^{0,2j-1,\alpha} \otimes a^{0,2j,\alpha}$.
$\underline{l \to l+1 \in [L-1]}$: Suppose the argument is true for $l \in [L-2]$.
Let $j \in [\frac{N}{2^{l+1}}], \gamma \in [r_{l+1}]$. Then, by definition we have that

$$\phi^{l+1,j,\gamma}_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{j\cdot 2^{l+1}}} = conv_{(l+1)}(j,\gamma)|_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{j\cdot 2^{l+1}}} =$$

$$= \langle a^{l+1,j,\gamma}, pool_{(l)}(j,:)\rangle|_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{j\cdot 2^{l+1}}} =$$

$$= \sum_{\alpha=1}^{r_l} a^{l+1,j,\gamma}_\alpha pool_{(l)}(j,\alpha)|_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{j\cdot 2^{l+1}}}$$

$$= \sum_{\alpha=1}^{r_l} a^{l+1,j,\gamma}_\alpha conv_{(l)}(2j-1,\alpha)\cdot conv_{(l)}(2j,\alpha)|_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{j\cdot 2^{l+1}}} =$$

$$= \sum_{\alpha=1}^{r_l} a^{l+1,j,\gamma}_\alpha (\phi^{l,2j-1,\alpha})_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{(j-1)\cdot 2^{l+1}+2^l}} \cdot (\phi^{l,2j,\alpha})_{d_{(j-1)\cdot 2^{l+1}+2^l+1},..,d_{j\cdot 2^{l+1}}} =$$

$$(\sum_{\alpha=1}^{r_l} a^{l+1,j,\gamma}_\alpha \phi^{l,2j-1,\alpha} \otimes \phi^{l,2j,\alpha})_{d_{(j-1)\cdot 2^{l+1}+1},..,d_{j\cdot 2^{l+1}}}$$

Thus, $\phi^{l+1,j,\gamma} = \sum_{\alpha=1}^{r_l} a^{l+1,j,\gamma}_\alpha \phi^{l,2j-1,\alpha} \otimes \phi^{l,2j,\alpha}$.
Applying the induction principal concludes the proof.
Finally, the output of the network is given by

$$\sum_{\alpha=1}^{r_{L-1}} a^L_\alpha conv_{(L-1)}(1,\alpha)\cdot conv_{(L-1)}(2,\alpha)|_{d_1,..,d_N} =$$

$$= \sum_{\alpha=1}^{r_{L-1}} a^L_\alpha (\phi^{L-1,1,\alpha})_{d_1,..,d_{\frac{N}{2}}} \cdot (\phi^{L-1,2,\alpha})_{d_{\frac{N}{2}+1},..,d_N} = (\sum_{\alpha=1}^{r_{L-1}} a^L_\alpha \phi^{L-1,1,\alpha} \otimes \phi^{L-1,2,\alpha})_{d_1,..,d_N}$$

As desired. □

3.2 Question 2

We'll denote the following sets of indices:

$$\mathcal{I} := \{i_1, .., i_{|\mathcal{I}|}\} \subset [n + \overline{n}] := [N]$$

$$\mathcal{I}^{\mathcal{C}} := \{\tilde{i}_1, .., \tilde{i}_{|\mathcal{I}^{\mathcal{C}}|}\}$$

$$\mathcal{J} := \mathcal{I} \cap [n] := \{j_1, .., j_{|\mathcal{J}|}\} \subset [n]$$

$$\mathcal{J}^{\mathcal{C}} := \{\tilde{j}_1, .., \tilde{j}_{|\mathcal{J}^{\mathcal{C}}|}\}$$

$$\mathcal{K} := (I - n) \cap [\overline{n}] := \{k_1, .., k_{|\mathcal{K}|}\} \subset [\overline{n}]$$

$$\mathcal{K}^{\mathcal{C}} := \{\tilde{k}_1, .., \tilde{k}_{|\mathcal{K}^{\mathcal{C}}|}\}$$

Thus, by definition of tensor matricization, for $\forall (d_1, .., d_N) \in \prod_{t=1}^{N}[m_t]$ the following holds:

$$(\llbracket T \otimes \overline{T} \rrbracket_{\mathcal{I}})_{1+\sum_{t=1}^{|\mathcal{I}|}(d_{i_t}-1)\prod_{t'=t+1}^{|\mathcal{I}|} m_{i_{t'}}, 1+\sum_{t=1}^{|\mathcal{I}^{\mathcal{C}}|}(d_{\tilde{i}_t}-1)\prod_{t'=t+1}^{|\mathcal{I}^{\mathcal{C}}|} m_{\tilde{i}_{t'}}} = (T \otimes \overline{T})_{d_1,..,d_N} = (*)$$

By definition of tensor outer product, we have that

$$(*) = T_{d_1,..,d_n} \cdot \overline{T}_{d_{n+1},..,d_N} =$$

$$(\llbracket T \rrbracket_{\mathcal{J}})_{1+\sum_{t=1}^{|\mathcal{J}|}(d_{j_t}-1)\prod_{t'=t+1}^{|\mathcal{J}|} m_{j_{t'}}, 1+\sum_{t=1}^{|\mathcal{J}^{\mathcal{C}}|}(d_{\tilde{j}_t}-1)\prod_{t'=t+1}^{|\mathcal{J}^{\mathcal{C}}|} m_{\tilde{j}_{t'}}} \cdot$$

$$\cdot (\llbracket \overline{T} \rrbracket_{\mathcal{K}})_{1+\sum_{t=1}^{|\mathcal{K}|}(d_{k_t}-1)\prod_{t'=t+1}^{|\mathcal{K}|} m_{k_{t'}}, 1+\sum_{t=1}^{|\mathcal{K}^{\mathcal{C}}|}(d_{\tilde{k}_t}-1)\prod_{t'=t+1}^{|\mathcal{K}^{\mathcal{C}}|} m_{\tilde{k}_{t'}}} =$$

$$= \begin{Bmatrix} (1+\sum_{t=1}^{|\mathcal{J}|}(d_{j_t}-1)\prod_{t'=t+1}^{|\mathcal{J}|} m_{j_{t'}}, 1+\sum_{t=1}^{|\mathcal{J}^{\mathcal{C}}|}(d_{\tilde{j}_t}-1)\prod_{t'=t+1}^{|\mathcal{J}^{\mathcal{C}}|} m_{\tilde{j}_{t'}}) =: (a, b) \\ (1+\sum_{t=1}^{|\mathcal{K}|}(d_{k_t}-1)\prod_{t'=t+1}^{|\mathcal{K}|} m_{k_{t'}}, 1+\sum_{t=1}^{|\mathcal{K}^{\mathcal{C}}|}(d_{\tilde{k}_t}-1)\prod_{t'=t+1}^{|\mathcal{K}^{\mathcal{C}}|} m_{\tilde{k}_{t'}}) =: (a', b') \end{Bmatrix} =$$

$$(\llbracket T \rrbracket_{\mathcal{J}})_{a,b} \cdot (\llbracket \overline{T} \rrbracket_{\mathcal{K}})_{a',b'} = ((\llbracket T \rrbracket_{\mathcal{J}})_{a,b} \cdot \llbracket \overline{T} \rrbracket_{\mathcal{K}})_{a',b'} =$$

$$= (\llbracket T \rrbracket_{\mathcal{J}} \odot \llbracket \overline{T} \rrbracket_{\mathcal{K}})_{(a-1)\cdot \#rows(\llbracket \overline{T} \rrbracket_{\mathcal{K}})+a', (b-1)\cdot \#cols(\llbracket \overline{T} \rrbracket_{\mathcal{K}})+b'} = (**)$$

The following holds:

$$(a-1) \cdot \#rows(\llbracket \overline{T} \rrbracket_{\mathcal{K}}) + a' =$$

$$= \sum_{t=1}^{|\mathcal{J}|}(d_{j_t}-1) \prod_{t'=t+1}^{|\mathcal{J}|} m_{j_{t'}} \prod_{t''=1}^{|\mathcal{K}|} m_{k_{t''}} + 1 + \sum_{t=1}^{|\mathcal{K}|}(d_{k_t}-1) \prod_{t'=t+1}^{|\mathcal{K}|} m_{k_{t'}} =$$

$$= 1 + \sum_{t=1}^{|\mathcal{I}|}(d_{i_t}-1) \prod_{t'=t+1}^{|\mathcal{I}|} m_{i_{t'}}$$

Moreover,

$$(b-1) \cdot \#cols(\llbracket \overline{T} \rrbracket_{\mathcal{K}}) + b' =$$

$$= \sum_{t=1}^{|\mathcal{J}^{\mathcal{C}}|}(d_{\tilde{j}_t}-1) \prod_{t'=t+1}^{|\mathcal{J}^{\mathcal{C}}|} m_{\tilde{j}_{t'}} \prod_{t''=1}^{|\mathcal{K}^{\mathcal{C}}|} m_{\tilde{k}_{t''}} + 1 + \sum_{t=1}^{|\mathcal{K}^{\mathcal{C}}|}(d_{\tilde{k}_t}-1) \prod_{t'=t+1}^{|\mathcal{K}^{\mathcal{C}}|} m_{\tilde{k}_{t'}} =$$

$$= 1 + \sum_{t=1}^{|\mathcal{I}^{\mathcal{C}}|} (d_{\tilde{i}_t} - 1) \prod_{t'=t+1}^{|\mathcal{I}^{\mathcal{C}}|} m_{\tilde{i}_{t'}}$$

Thus,

$$(**) = (\llbracket T \rrbracket_{\mathcal{J}} \odot \llbracket \overline{T} \rrbracket_{\mathcal{K}})_{1+\sum_{t=1}^{|\mathcal{I}|}(d_{i_t}-1)\prod_{t'=t+1}^{|\mathcal{I}|} m_{i_{t'}}, 1+\sum_{t=1}^{|\mathcal{I}^{\mathcal{C}}|}(d_{\tilde{i}_t}-1)\prod_{t'=t+1}^{|\mathcal{I}^{\mathcal{C}}|} m_{\tilde{i}_{t'}}}$$

Therefore, $\llbracket T \otimes \overline{T} \rrbracket_{\mathcal{I}}$ and $\llbracket T \rrbracket_{\mathcal{J}} \odot \llbracket \overline{T} \rrbracket_{\mathcal{K}} = \llbracket T \rrbracket_{\mathcal{I} \cap [n]} \odot \llbracket \overline{T} \rrbracket_{(\mathcal{I}-n) \cap [\overline{n}]}$ coincide on every entry, making the matricizations equal as desired. □

3.3 Question 3

To establish condition (i) of the expressive efficiency argument, we will follow a similar path to that of the original case.
Suppose the shallow network realizes a function with the convolutional filter weights $\{(a^{z,1}, ..., a^{z,N}) \in (\mathbb{R}^M)^N\}_{z=1}^Z$ and the output weights $a \in \mathbb{R}^Z$.
We'll construct the same function in the deep network:
0. The network will be of width Z across all of its layers
1. The output weights will be set to a
2. The convolutional filter weights of the first layer will be set to

$$\{(a^{0,1,\gamma}, .., a^{0,N,\gamma})\}_{\gamma=1}^Z = \{(a^{z,1}, ..., a^{z,N}) \in (\mathbb{R}^M)^N\}_{z=1}^Z$$

3. The rest of the convolutional filter weights will be set $a^{l,j,\gamma} = e^\gamma$ for layers 1 to $L-1$, all indices j in every layer, and all channels γ in every layer
Step 3 ensures that all layers above 0 are passthrough.
The pooling windows of size 4 with the combination of the last argument ensure global pooling.
Combined with the same output weights we have that the deep network realizes the same function as the shallow network.
We will procced to establish condition (ii). Applying the linearity of matrices and the fact that the matricization of outer products is the Kronecker product of the matricizations, the following holds for the CP decomposition of the shallow network:

$$\llbracket A^{CP} \rrbracket = \llbracket \sum_{z=1}^Z a_z a^{1,z} \otimes .. \otimes a^{N,z} \rrbracket = \sum_{z=1}^Z a_z \llbracket a^{1,z} \otimes .. \otimes a^{N,z} \rrbracket =$$

$$= \sum_{z=1}^Z a_z \llbracket a^{1,z} \otimes a^{2,z} \rrbracket \odot .. \odot \llbracket a^{N-1,z} \otimes a^{N,z} \rrbracket$$

In each summand, each matricization is of rank of no more than 1 since its an outer product of two vectors.
Thus, applying the rank property of Kronecker products, we have that

$$rank(\llbracket A^{CP} \rrbracket) = rank(\sum_{z=1}^Z a_z \llbracket a^{1,z} \otimes a^{2,z} \rrbracket \odot .. \odot \llbracket a^{N-1,z} \otimes a^{N,z} \rrbracket) \leq \sum_{z=1}^Z 1 \cdot .. \cdot 1 = Z$$

To establish condition (ii), we will follow similar steps to the ones seen in class. We will assume that $r_0 \geq M$. We will constuct the following function in the deep network:
1. The convolutional filter weights of layer 0 will be

$$a^{0,j,\gamma} = \begin{cases} e^{\gamma} & \gamma \leq M \\ 0_{M\times 1} & else \end{cases}$$

2. The convolutional filter weights of layer 1 will be

$$a^{1,j,\gamma} = \begin{cases} 1_{r_0\times 1} & \gamma = 1 \\ 0_{r_0\times 1} & else \end{cases}$$

3. The convolutional filter weights of layers $l \in \{2, .., L-1\}$ will be

$$a^{l,j,\gamma} = \begin{cases} e^{1} & \gamma = 1 \\ 0_{r_{l-1}\times 1} & else \end{cases}$$

4. The output weights will be $a_L = e^1$
The HT decomposition we saw can be extended with the exact steps we saw in class to the case where pooling is done in windows of size 4, and N is a power of 4.
Thus, we have that

$$[\![\phi^{1,j,1}]\!] = [\![\sum_{\alpha=1}^{M} e^{\alpha} \otimes e^{\alpha} \otimes e^{\alpha} \otimes e^{\alpha}]\!] = I_{M\times M} \odot I_{M\times M}$$

Therefore, the matricization of the (extended) HT decomposition of the function of the entire network takes the form of

$$[\![A^{HT}]\!] = \bigodot_{k=1}^{log_4 \frac{N}{4}} (I_{M\times M} \odot I_{M\times M})$$

Thus, applying the rank property of the Kronecker product we have that

$$rank([\![A^{HT}]\!]) = rank(\bigodot_{k=1}^{\frac{N}{4}}(I_{M\times M} \odot I_{M\times M})) = (M^2)^{\frac{N}{4}} = M^{\frac{N}{2}}$$

Thus, in order for the shallow network to realize the function we constructed, $B = Z$ must be no lesser than $M^{\frac{N}{2}}$ - exponential in N. $\square$
To prove that the specific function realized above by the deep network is inapproximable by the shallow network unless B is exponential w.r.t N, we can use the same argument we saw in class for the original configuration.
Namely, this can be reached by analyzing the Frobenius distance between the matricizations of the tensor of the specific function above, and the tensor of any

function realized by the shallow network.
To prove the complete expressive efficiency of the deep network over the shallow network, it can be shown, in an analogous way to what was shown in class, that for almost any weight assignment in the deep network, the matricization of the tensor of the result function has rank which is exponential w.r.t N, and thus it can't be realized by the shallow network.
p.s. - We didn't fully prove inapproximablity and completeness since we weren't sure if it was needed, and thus only included sketches.

3.4 Question 4

According to the HT decomposition, we have that

$$\forall \alpha \in [r_{L-1}].[\![\phi^{L-1,1,\alpha}]\!]_{quad} = \sum_{\alpha'=1}^{r_{L-2}} a_{\alpha'}^{L-1,1,\alpha} [\![\phi^{L-2,1,\alpha'}]\!]_{quad} \odot [\![\phi^{L-2,2,\alpha'}]\!]_{quad}$$

$[\![\phi^{L-2,1,\alpha'}]\!]_{quad}$ is a column vector (since we matricize on all of its axis). On the other hand, we have from the opposite argument that $[\![\phi^{L-2,2,\alpha'}]\!]_{quad}$ is a row vector. Thus, we have that

$$\forall \alpha \in [r_{L-1}].rank([\![\phi^{L-1,1,\alpha}]\!]_{quad}) \leq r_{L-2}$$

The exact proof also holds for the $\phi^{L-1,2,\alpha}$ and thus

$$\forall \alpha \in [r_{L-1}].rank([\![\phi^{L-1,2,\alpha}]\!]_{quad}) \leq r_{L-2}$$

For the output tensor, we have that

$$[\![A^{HT}]\!]_{quad} = \sum_{\alpha=1}^{r_{L-1}} a_{\alpha}^{L} [\![\phi^{L-1,1,\alpha}]\!]_{quad} \odot [\![\phi^{L-1,2,\alpha}]\!]_{quad}$$

Thus, we receive that

$$rank([\![A^{HT}]\!]_{quad}) \leq r_{L-1} r_{L-2}^2$$

We saw in class that the separation rank of a function is equal to the rank of its corresponding tensor's matricization w.r.t to the partition.
Thus, we have that for any function f realized by the deep network

$$sep(f; quad) = rank([\![A^{HT}]\!]_{quad}) \leq r_{L-1} r_{L-2}^2 . \square$$