Appendix of Learning to Select Base Classes for Few-shot Classification

1. Proof of the Main Theories

1.1. Proof for Corollary 4.1

Lemma 1. *$\forall n \in N, g_n : 2^{B_u} \to \mathbb{R}_{\geq 0}, g_n(U) := M^K(f(c_n, \{c_{B_s}, c_U\}))$ is a submodular function.*

Proof. $\forall A \subseteq B \subseteq B_U$, let $u \in B_U \backslash B$, define the top-K similar classes with class n in $B_s \cup A$ and $B_s \cup B$ are K_A and K_B separately, we also define that after adding class u to both A and B, the top-K similar classes become K_A^u and K_B^u. Next, we discuss four cases:

(1) $u \in K_A^u$ but $u \in K_B^u$: In this case, $g_n(A+u) - g_n(A) = f(c_n, c_u) - \min_{x \in K_A} f(c_n, c_x)$ and $g_n(B+u) - g_n(B) = f(c_n, c_u) - \min_{x \in K_B} f(c_n, c_x)$. As $(B_s \cup A) \subseteq (B_s \cup B)$, there must be $\min_{x \in K_A} f(c_n, c_x) \leq \min_{x \in K_B} f(c_n, c_x)$. Thus we have $g_n(A+u) - g_n(A) \geq g_n(B+u) - g_n(B)$.

(2) $u \in K_A^u$ but $u \notin K_B^u$: In this case, easy to show that $g_n(A+u) - g_n(A) > 0 = g_n(B+u) - g_n(B)$.

(3) $u \notin K_A^u$ but $u \in K_B^u$: This case will not exist, as it represents that $f(c_n, c_u) \leq \min_{x \in K_A} f(c_n, c_x)$ and $f(c_n, c_u) \geq \min_{x \in K_B} f(c_n, c_x)$. This will induce a contradictory to $\min_{x \in K_A} f(c_n, c_x) \leq \min_{x \in K_B} f(c_n, c_x)$.

(4) $u \notin K_A^u$ but $u \notin K_B^u$: In this case, easy to show that $g_n(A+u) - g_n(A) = g_n(B+u) - g_n(B) = 0$.

In conclusion, $\forall A \subseteq B \subseteq B_U$, $u \in B_U \backslash B$, we have $g_n(A+u) - g_n(A) \geq g_n(B+u) - g_n(B)$, which demonstrates that $g_n(\cdot)$ is a submodular function. □

Corollary 1. ***(Corollary 4.1 in original paper)** Considering optimization problem 3 (in original paper), when $\lambda = 0$, Problem 3 is equivalent to a submodular monotone non-decreasing optimization with exact cardinality constraint and when $\lambda > 0$, Problem 3 is equivalent to a submodular optimization with exact cardinality.*

Proof. When $\lambda = 0$, by Lemma 1 and the property of the additivity of submodular function that if f and g are both submodular, then $h = f + g$ is also submodular, easy to show that the optimization function is submodular. Easy to show that $g_n(U)$ is also monotone non-decreasing, so Problem 3 with $\lambda = 0$ is a submodular monotone non-decreasing optimization with exact cardinality constraint. Also, the regularizer term of the optimization function

$$R(U) = \sum_{n \in N} \frac{1}{|B_s| + m} \sum_{u \in B_s \cup U} f(c_n, c_u)$$

is a modular function satisfying $R(A+u) - R(A) = R(B+u) - R(B), \forall A \subseteq B \subseteq V$. By the property of submodular function, the whole optimization function 3 with $\lambda > 0$ is also a submodular function (but not monotone non-decreasing). □

1.2. Proof for Theorem 1

Theorem 1. *For $B_s = \emptyset$ and $\lambda = 0$, when $m \geq K \cdot |N|$, using Greedy on Novel Class to solve for optimization problem 3, the solution will be optimal.*

Proof. The maximum number of base classes for top-K most similar classes with each novel class is $K \cdot |N|$, thus when $m \geq K \cdot |N|$, a greedy algorithm on finding top-K most similar classes for each novel class is optimal. □

1.3. Proof for Theorem 2

Theorem 2. *For $B_s = \emptyset$ and $\lambda = 0$, using Greedy on Target Function to solve for optimization problem 3, let $h(\cdot)$ be the optimization function, and let Q be*

$$Q = \mathbb{E}_{u \sim Uniform(B), v \sim Uniform(N)}(f(c_u, c_v))$$

representing for the average similarity between base classes and novel classes, we have $h(U) \geq (1 - 1/e) \cdot h(OPT) + 1/e \cdot Q$.

Proof. Let us suppose A_i denotes the chosen subset after greedy step i. Let function $\gamma(u) = \frac{1}{N}\sum_{n\in N} f(c_n, c_u)$. According to the greedy algorithm, A_K should be top-k elements in B_u maximizing $\gamma(u)$. Easy to show that $h(A_K) = \frac{1}{K}\sum_{u\in A_K}\gamma(u) \geq Q$.

[3] shows that for submodular monotone non-decreasing problem, we have

$$h(OPT) - h(A_i) \leq (1 - 1/k)\cdot(h(OPT) - h(A_{i-1})),$$

Combining the inequality for every $K \leq i \leq m$ and take limitations we have

$$h(U) = h(A_m) \geq (1 - 1/e)\cdot h(OPT) + 1/e \cdot h(A_K) \geq (1 - 1/e)\cdot h(OPT) + 1/e\cdot Q.$$

□

1.4. Proof for Theorem 3

Theorem 3. *Let $S \subseteq B_u$ is a random set, with each element v in B_u i.i.d sampled with probability $(x \wedge (B_u - u))_v$. For each novel class $n \in N$, we sort the similarity function $f(c_n, c_b)$ for every base class $b \in B$ in descent order, denoting as $q_{n,[1]}, q_{n,[2]}, \cdots q_{n,[|B|]}$. Similarly, we also sort the similarity function for every base class in $S \cup B_s$ in descent order, denoting as $s_{n,[1]}, s_{n,[2]}, \cdots s_{n,[|S|+|B_s|]}$, then we have:*

$$\begin{aligned}
&F(x\vee u) - F(x\wedge(B_u - u))\\
&= \frac{1}{|N|\cdot K}\sum_{n\in N}\sum_{i=1}^{|B|} P(s_{n,[K]} = q_{n,[i]})\max(f(c_n,c_u) - q_{n,[i]}, 0)\\
&- \lambda\cdot\frac{1}{|N|\cdot m}\sum_{n\in N} f(c_n, c_u)
\end{aligned} \tag{1}$$

Proof.

$$\begin{aligned}
&F(x\vee u) - F(x\wedge(B_u - u))\\
&= \sum_{S\subseteq B_u\backslash u} h(S+u)\prod_{\substack{v\in S\\ v\neq u}} x_v \prod_{\substack{v\notin S\\ v\neq u}}(1 - x_v)\cdot 1 - \sum_{S\subseteq B_u\backslash u} h(S)\prod_{\substack{v\in S\\ v\neq u}} x_v\prod_{\substack{v\notin S\\ v\neq u}}(1 - x_v)\cdot 1\\
&= \sum_{S\subseteq B_u\backslash u}(h(S+u) - h(S))\prod_{\substack{v\in S\\ v\neq u}} x_v\prod_{\substack{v\notin S\\ v\neq u}}(1 - x_v)\\
&= \sum_{S\subseteq B_u\backslash u}(\frac{1}{|N|\cdot K}\sum_{n\in N}\max(f(c_n,c_u) - s_{n,[K]}, 0))\prod_{\substack{v\in S\\ v\neq u}} x_v\prod_{\substack{v\notin S\\ v\neq u}}(1 - x_v)\\
&\quad - \lambda\cdot\frac{1}{|N|\cdot m}\sum_{n\in N} f(c_n,c_u)\sum_{S\subseteq B_u\backslash u}(\prod_{\substack{v\in S\\ v\neq u}} x_v\prod_{\substack{v\notin S\\ v\neq u}}(1 - x_v))\\
&= \frac{1}{|N|\cdot K}\sum_{n\in N}\sum_{i=1}^{|B|} P(s_{n,[K]} = q_{n,[i]})\max(f(c_n,c_u) - q_{n,[i]}, 0)\\
&\quad - \lambda\cdot\frac{1}{|N|\cdot m}\sum_{n\in N} f(c_n,c_u)
\end{aligned}$$

□

1.5. Proof for Equation 7 (in Original Paper)

The only unknown term $P(s_{n,[K]} = q_{n,[i]})$ for $n \in N$ could be solved using dynamic programming in $O(K\cdot|B|\cdot|N|)$ time complexity by the following two recursion equations:

$$\begin{cases} P(s_{n,[j]} \geq q_{n,[i]}) = (1 - x_{[i]})\cdot P(s_{n,[j]} \geq q_{n,[i-1]}) + x_{[i]}\cdot P(s_{n,[j-1]} \geq q_{n,[i-1]})\ for\ [i] \in B_u\\ P(s_{n,[j]} \geq q_{n,[i]}) = P(s_{n,[j-1]} \geq q_{n,[i-1]})\ for\ [i] \in B_s\end{cases} \tag{2}$$

$$P(s_{n,[j]} = q_{n,[i]}) = P(s_{n,[j]} \geq q_{n,[i]}) - P(s_{n,[j]} \geq q_{n,[i-1]}) \tag{3}$$

Proof. Equation 3 is obvious. Below we give the proof of 2, for the case of $[i] \in B_u$:

$$\begin{aligned}
&P(s_{n,[j]} = q_{n,[i]}) \\
&= \{P(s_{n,[j-1]} \geq q_{n,[i-1]}) - \sum_{m=1}^{i-1} P(s_{n,[j]} = q_{n,[m]})\} \cdot x_{[i]} \\
&= \{P(s_{n,[j-1]} \geq q_{n,[i-1]}) - \sum_{m=1}^{i-1} (P(s_{n,[j]} \geq q_{n,[m]}) - P(s_{n,[j]} \geq q_{n,[m-1]}))\} \cdot x_{[i]} \\
&= \{P(s_{n,[j-1]} \geq q_{n,[i-1]}) - P(s_{n,[j]} \geq q_{n,[i-1]})\} \cdot x_{[i]}
\end{aligned}$$

Plug Equation 3 to the equation above and that will be Equation 2.

Note that the vector x could also be seen as an extension form in $[0,1]^{|B|}$: $x_{[i]}$ represents for the probability of element $[i]$ being selected, when using these two equations, if $[i] \in B_s$ we set $x_{[i]} = 1$; if $[i] = v \in B_u \backslash u$, we set $x_{[i]} = x_v$; and if $[i] = u$ we set $x_{[i]} = 0$. Hence for $[i] \in B_s$ we have $x_{[i]} = 1$ and plug into the first equation of 2 to obtain the second equation. □

1.6. Proof for Theorem 4

Theorem 4. *For $B_s = \emptyset$ and $\lambda > 0$, using a combination of Random Greedy Algorithm and Continuous Double Greedy Algorithm to solve for optimization problem 3, let $h(\cdot)$ be the optimization function, and let Q be*

$$Q = \mathbb{E}_{u \sim Uniform(B), v \sim Uniform(N)}(f(c_u, c_v))$$

representing for the average similarity between base classes and novel classes, and let r be the cardinality of B_u, we have

$$\mathbb{E}(h(U)) \geq max(\frac{1 - m/er}{e} \cdot h(OPT) + C_1 \cdot Q, (1 + \frac{r}{2\sqrt{(r-m)m}})^{-1} \cdot h(OPT) + C_2 \cdot Q)$$

For $0 < \lambda < \frac{1}{e-1}$, we have $C_1 = \frac{1}{e} + (1 - \frac{1}{e})\frac{m}{r} - (1 - \frac{1}{e}) \cdot \lambda > 0$ and $C_2 = \frac{(1-\lambda)r}{2\sqrt{(r-m)m+r}} - \epsilon \geq \frac{1}{2}(1-\lambda) > 0$.

Proof. 1. For random greedy algorithm, our proof follows the Lemma 4.7 and Lemma 4.8 in [2] with slight differences. We suggest the readers read the proof of [2] foreahead. The first difference is that $h(B_u)$ may be negative, and it should not be taken away while calculating $\mathbb{E}(h(A_{i-1} \cup M_i'))$. Considering $h(B_u) < 0$ we have:

$$\frac{m - X_{i-1}}{r} \cdot h(B_u) \geq \frac{m}{r} \cdot h(B_u) = \frac{m}{r}(1 - \lambda \cdot \frac{r}{m}) \cdot Q \tag{4}$$

The inequality follows by the definition of X_{i-1}: $X_{i-1} = |OPT \backslash A_i| \geq 0$. And this term should be added to RHS of Lemma 4.7 in [2] with a m^{-1} coefficient according to the process of proof. Thus Lemma 4.7 could be rewritten in our problem as: for every $K \leq i \leq m$:

$$\begin{aligned}
\mathbb{E}(h_{u_i}(A_{i-1})) \geq &\frac{[r/m - 1 + (1 - 1/m)^{i-1}] \cdot (1 - 1/k)^{i-1}}{n} \cdot h(OPT) \\
&- \frac{\mathbb{E}(h(A_{i-1})}{k} + \frac{1}{r}(1 - \lambda \cdot \frac{r}{m}) \cdot Q - E_{i-1}
\end{aligned}$$

The second difference is that we start our algorithm from $i = K$ and similar to Theorem 1 we have

$$h(A_K) = (\frac{1}{K} - \frac{\lambda}{m}) \sum_{u \in A_K} \gamma(u) \geq (1 - \frac{\lambda \cdot K}{m}) \cdot Q. \tag{5}$$

Thus compared to Lemma 4.8 in [2], we need to add two terms related to Equation 4 and 5. After repeated applications of Lemma 4.7, the term related to 4 is calculated by:

$$\lim_{\substack{K \to 0 \\ m \to +\infty}} \sum_{i=K}^{m} (1 - \frac{1}{m})^i \cdot (\frac{1}{r}(1 - \lambda \cdot \frac{r}{m}) \cdot Q) = (1 - \frac{1}{e})\frac{m}{r}(1 - \lambda\frac{r}{m}) \cdot Q.$$

And the term related to 5 is calculated by:

$$\lim_{\substack{K\to 0\\ m\to +\infty}} (1-\frac{1}{m})^{m-K}h(A_k) \geq \frac{1}{e}\cdot Q.$$

Thus combine these term with the coefficient $h(OPT)$ unchanged we could prove that:

$$C_1 = \frac{1}{e} + (1-\frac{1}{e})\frac{m}{r} - (1-\frac{1}{e})\cdot\lambda$$

And for $\lambda > 1/(e-1)$, C_1 guarantees to be non-negative.

2. For double continuous greedy algorithm, refer to the Theorem 3.2 in [2] with some deformation we have:

$$h(U) \geq \frac{h(OPT)+\frac{1}{2}(\sqrt{\frac{r-m+K}{m-K}})h(A_K)+\sqrt{\frac{m-K}{r-m+K}}h(B_u))}{1+\frac{1}{2}\frac{r}{\sqrt{(r-m+K)(m-K)}}} \tag{6}$$

From Theorem 1, we could conclude that $h(A_K) = (\frac{1}{K}-\frac{\lambda}{m})\sum_{u\in A_K}\gamma(u) \geq (1-\frac{\lambda\cdot K}{m})\cdot Q$. Also, easy to show that $h(B_u) \geq (1-\frac{\lambda\cdot r}{m})\cdot Q$. Thus we could put these two inequality to Equation 6 and as $K << m$ and $K << r$, we could omit the term with K. Then Equation 6 is equivalent to the inequality below:

$$h(U) \geq (1+\frac{r}{2\sqrt{(r-m)m}})^{-1}\cdot h(OPT) + C_2\cdot Q) \tag{7}$$

And we have:

$$\begin{aligned} C_2 &= (1+\frac{r}{2\sqrt{(r-m)m}})^{-1}\cdot\frac{1}{2}\cdot(\sqrt{\frac{r-m}{m}}+\sqrt{\frac{m}{r-m}}-\frac{\lambda\cdot r}{\sqrt{(r-m)m}})-\epsilon \\ &= \frac{(1-\lambda)r}{2\sqrt{(r-m)m}+r}-\epsilon \end{aligned}$$

Let $\alpha = m/r \in (0,1)$ denote for the proportion of chosen classes with respect to all classes, we find that $C = (1+2\sqrt{(1-\alpha)\alpha})^{-1}(1-\lambda)$. Thus, the extremum is taken at $\alpha = 1/2$, and we have $C \geq \frac{1}{2}\cdot(1-\lambda)$, which is our theorem. □

Theorem 2 shows that when combining random greedy algorithm and double continuous greedy algorithm, and for $0 < \lambda < 1/(e-1)$, we could reach a 0.356-approximation. It could be easily shown by comparing the two bounds that when $m < 0.082r$ or $m > 0.918r$ we choose random greedy algorithm and when $0.082r \leq m \leq 0.918r$ we choose double continuous greedy algorithm.

2. Details of Continuous Double Greedy Algorithm

2.1. Reduction

To simplify our discussion in the original paper, we assume the following reduction of the original problem [2] is applied:

Reduction 1. *For the problem of* $max\ \{h(U) : |U| = m, U \subset B_u\}$*, we may assume* $2m < |B_u|$.

Proof. If this is not the case, let $\bar{m} = |B_u| - m$ and $\bar{h}(U) = h(B_u\backslash U)$, it could be easily checked that $2\bar{m} < |B_u|$ and the problem $max\ \{\bar{h}(U) : |U| = \bar{m}, U \subset B_u\}$ is equivalent to the original problem. □

The details of Algorithm 4 in the original paper are based on the assumption $2m \leq |B_u|$.

2.2. Initial State of Dynamic Programming

The explanation for $P(s_{n,[j]} \geq q_{n,[i]})$ is the probability of the jth-largest similarity between base classes in the random set $S \in B_s$ and the novel class n larger than $q_{n,[i]}$, *i.e.* the ith-largest similarity between base classes in B_u and the novel class n. From this definition, the initial state of the dynamic programming process is:

$$P(s_{n,[1]} \geq q_{n,[1]}) = x_{[1]}$$

$$P(s_{n,[j]} \geq q_{n,[1]}) = 0 \quad for \quad j = 2,3,\cdots K$$

$$P(s_{n,[1]} \geq q_{n,[i]}) = (1-P(s_{n,[1]} \geq q_{n,[i-1]}))\cdot x_{[i]} \quad for \quad i = 2,3,\cdots |B|$$

2.3. Pruning of Dynamic Programming

According to Equation 6 and 7 in original paper, we need to calculate $P_u(s_{n,[j]} \geq q_{n,[i]})$ for $j = 1 \cdots K$ and $i = 1 \cdots |B|$, for each $u \in B_u$ and novel class n. Noticing that here we use P_u instead of P because for each $u \in B_u$, we must set $x_u = (x \wedge (B_u - u))_u = 0$ and run dynamic programming by Equation 7. Thus the result for $P(s_{n,[j]} \geq q_{n,[i]})$ is different considering selecting different u. Traditionally, we need to fix and loop $u \in B_u$, $n \in N$ to calculate $P_u(s_{n,[j]} \geq q_{n,[i]})$ in $O(K \cdot |B|^2 \cdot |N|)$. However, in this section, we introduce a pruning method, which could largely decrease the time complexity.

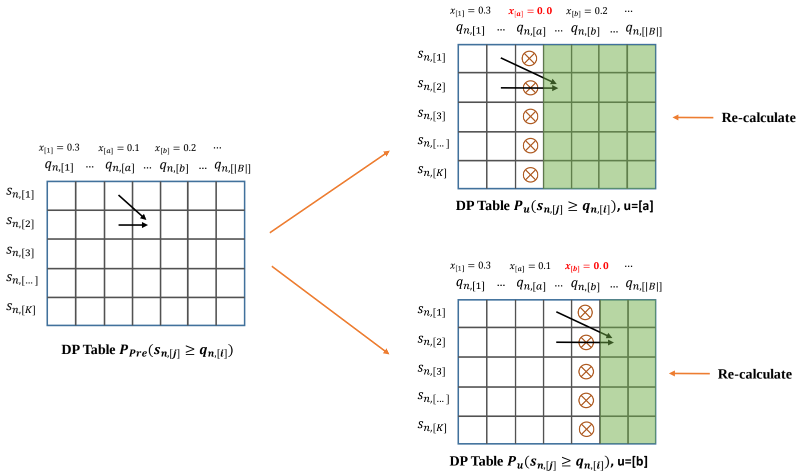
$x_{[1]} = 0.3$ $x_{[a]} = 0.1$ $x_{[b]} = 0.2$ ⋯
$q_{n,[1]}$ ⋯ $q_{n,[a]}$ ⋯ $q_{n,[b]}$ ⋯ $q_{n,[|B|]}$
$s_{n,[1]}$
$s_{n,[2]}$
$s_{n,[3]}$
$s_{n,[\dots]}$
$s_{n,[K]}$
DP Table $P_{Pre}(s_{n,[j]} \geq q_{n,[i]})$
$x_{[1]} = 0.3$ $x_{[a]} = 0.0$ $x_{[b]} = 0.2$ ⋯
$q_{n,[1]}$ ⋯ $q_{n,[a]}$ ⋯ $q_{n,[b]}$ ⋯ $q_{n,[|B|]}$
$s_{n,[1]}$
$s_{n,[2]}$
$s_{n,[3]}$
$s_{n,[\dots]}$
$s_{n,[K]}$
Re-calculate
DP Table $P_u(s_{n,[j]} \geq q_{n,[i]})$, u=[a]
$x_{[1]} = 0.3$ $x_{[a]} = 0.1$ $x_{[b]} = 0.0$ ⋯
$q_{n,[1]}$ ⋯ $q_{n,[a]}$ ⋯ $q_{n,[b]}$ ⋯ $q_{n,[|B|]}$
$s_{n,[1]}$
$s_{n,[2]}$
$s_{n,[3]}$
$s_{n,[\dots]}$
$s_{n,[K]}$
Re-calculate
DP Table $P_u(s_{n,[j]} \geq q_{n,[i]})$, u=[b]

Figure 1. Example of Pruning Methods

The keypoint is that we could pre-compute $P_{pre}(s_{n,[j]} \geq q_{n,[i]})$, as Figure 1 shows. The dynamic programming (DP) table of $P_{pre}(s_{n,[j]} \geq q_{n,[i]})$ is constructed by setting the probability vector x as its original value, without setting certain x_u to be 0. In this way, when different $u \in B_u$ is selected, we could utilize this pre-calculation DP table. The unique difference for calculating $P_{pre}(s_{n,[j]} \geq q_{n,[i]})$ and $P_u(s_{n,[j]} \geq q_{n,[i]})$ is that we need to set corresponding $x_u = 0$, as the right part of Figure 1 shows. Let us suppose $u = [a]$ and we encourage two pruning methods in this paper: First, if $|P_{pre}(s_{n,[j]} \geq q_{n,[a]}) - P_{pre}(s_{n,[j]} \geq q_{n,[a-1]})| < \epsilon$ for all $j = 1, \cdots, K$, then there is no need to re-calculate $P_u(s_{n,[j]} \geq q_{n,[i]})$, and we could directly use $P_{pre}(s_{n,[j]} \geq q_{n,[i]})$ instead. Noticing that when a is relatively large, there is a high probability satisfying the condition above, thus we could decrease the constant number of the time complexity of the algorithm substantially. Second, even though there is need to re-calculate DP table for $P_u(s_{n,[j]} \geq q_{n,[i]})$, we find that the left part of the DP table of column a does not need to re-calculate as well, as Figure 1 shows. In this way, we could only re-calculate the right part (the green area). By using these two pruning methods simultaneously, the general complexity of the algorithm is relatively low compared with the worst case $O(T \cdot K \cdot |B|^2 \cdot |N|)$. The algorithm could further be easily extended to parallel computing version for a greater acceleration.

2.4. Pipage Rounding

The original paper mentions that, to transform the fractional solution obtained by Algorithm 4 to an integral solution, we may use some rounding techniques. One of the classical trick is Pipage Rounding.

We need three things to make Pipage Rounding work:

1. For any $x \in \mathcal{P}$, we need a vector v and $\alpha, \beta > 0$ such that $x + \alpha v \in \mathcal{P}$ or $x - \beta v \in \mathcal{P}$ have strictly more integral coordinates.

2. For all x, the function $g_x(t) := F(x + tv)$ needs to be convex.

3. Finally, we need a starting fractional x with a guarantee that $F(x) \geq \rho \cdot OPT$.

where $\mathcal{P} = \{x \in [0,1]^{|B_u|} : \sum_{j=1}^{|B_u|} x_j = m\}$ is a polytope constraint and $F(\cdot)$ is the multi-linear extension of the original optimization function $h(\cdot)$.

Noticing that the assumption 2 and 3 are satisfied in Non-monotone Submodular Optimization, where assumption 2 is proved by [1], and assumption 3 is consistent with Theorem 4 in original paper. Next we focus on assumption 1.

Suppose x is a non-integral vector in $\mathcal{P}$ and there are at least two fractional coordinates. Let it be x_p and x_q. Define $v = e_p - e_q$, where e_p is the vector with 1 in the pth coordinate and 0 elsewhere. Let $\alpha = min(1 - x_p, x_q)$ and $\beta = min(1 - x_q, x_p)$. After constructing v, α, β, easy to show that $x + \alpha v$ and $x - \beta v$ are both in $\mathcal{P}$ and both of them have strictly more integral coordinates.

We show that all three assumptions are satisfied, for running Pipage Rounding, we select two coordinates of x at each time, selecting v, α, β as above, compare $F(x + \alpha v)$ and $F(x - \beta v)$ and pick the probability vector making the value larger (*i.e.* $x + \alpha v$ or $x - \beta v$) as the new probability vector x. When calculating the value of the function, we still just need to calculate $F(x + \alpha v) - F(x)$ instead of directly calculating $F(x + \alpha v)$ by the equation:

$$\begin{aligned}
&F(\cdots, 1, \cdots, x_q', \cdots) - F(\cdots, x_p, \cdots, x_q, \cdots) = \\
&\quad (F(\cdots, 1, \cdots, x_q', \cdots) - F(\cdots, x_p, \cdots, x_q', \cdots)) + \\
&\quad (F(\cdots, x_p, \cdots, x_q', \cdots) - F(\cdots, x_p, \cdots, x_q, \cdots)),
\end{aligned}$$

and convert the problem of change in two coordinates to change in only one coordinate, which could be solved using dynamic programming the same as Equation 6 and 7 with a slight difference, as is the case of $F(x - \beta v)$. Repeat this process until the component of x is all integral (*i.e.* 1 or 0). From assumption 1 we know that the algorithm will definitely converged to an integral solution. [1] also shows that the integral solution x^* after running Pipage Rounding also satisfies $F(x^*) \geq F(x)$, which does not change the lower bound of Continuous Double Greedy Algorithm.

2.5. Extensions

We also note that Algorithm 4 (along with Algorithm 3) in original paper has more applicability in real cases, especially when there are some modular constraints. For example, we could add a constraint to the original problem that the difficulty of obtaining a sufficient image set for each base class could be quantified as a real number, and we should balance the accuracy of classification on novel classes and the total difficulty of obtaining base dataset when selecting base classes. The setting is equivalent to substact a hyper-parameter μ multiplying the total difficulty from the original optimization function. Noticing the total difficulty is a modular term, thus the new optimization function is also submodular and we could still solve this new problem by non-monotone submodular optimization.

3. Complexity Analysis

For Algorithm 1, we use a balanced binary search tree to record the similarity of base classes with each novel class. Establishing and updating the search tree cost $O(|B| \cdot log|B| \cdot |N|)$ totally.

For Algorithm 2 and 3, we use a minimum heap to record current top-K similar base classes for each novel class. For each turn, the calculation of all $h(u_i|U_{i-1})$ costs $O(|B|)$, for Algorithm 2 finding the top-1 of $h(u_i|U_{i-1})$ costs $O(|B|)$ and for Algorithm 3 finding top-m elements costs $O(|B| \cdot logm)$. Finally the update of the minimum heap costs $O(|N| \cdot logK)$. Thus totally the complexity is $O(m \cdot (|B| + |N| \cdot logK))$ for Algorithm 2 and $O(m \cdot (|B| \cdot logm + |N| \cdot logK))$ for Algorithm 3.

For Algorithm 4, for each turn t and for each $u \in B_u$, the dynamic programming process costs $O(K \cdot |B| \cdot |N|)$. Thus, the worst-case complexity of the Double Continuous Greedy Algorithm is $O(T \cdot K \cdot |B|^2 \cdot |N|)$. However, with some pruning strategy (see Appendix), the constant number of the complexity is relatively low (much smaller than 1).

4. More Ablation Studies

4.1. Effects of Model Head

Table 1. ImageNet: Pre-trained Selection, 100-way novel classes

Algorithm	Head	m=100, 5-shot	m=100, 20-shot	m=20, 5-shot	m=20, 20-shot
Random	1-NN	$39.39\% \pm 0.82\%$	$49.47\% \pm 0.67\%$	$23.89\% \pm 0.56\%$	$33.06\% \pm 0.47\%$
	SR	$38.74\% \pm 0.76\%$	$50.20\% \pm 0.40\%$	$24.29\% \pm 0.49\%$	$36.38\% \pm 0.37\%$
DomSim	1-NN	$38.00\% \pm 0.36\%$	$48.80\% \pm 0.79\%$	$23.15\% \pm 0.43\%$	$31.81\% \pm 0.58\%$
	SR	$38.84\% \pm 0.74\%$	$52.81\% \pm 0.20\%$	$23.62\% \pm 0.29\%$	$36.31\% \pm 0.45\%$
Alg. 1, $K = 1, \lambda = 0$	1-NN	$43.42\% \pm 0.78\%$	$53.79\% \pm 0.37\%$	$25.71\% \pm 0.43\%$	$34.67\% \pm 0.36\%$
	SR	$\mathbf{43.72\% \pm 0.47\%}$	$55.84\% \pm 0.38\%$	$26.08\% \pm 0.45\%$	$37.75\% \pm 0.22\%$
Alg. 2, $K = 1, \lambda = 0$	1-NN	$43.20\% \pm 0.76\%$	$53.61\% \pm 0.27\%$	$26.13\% \pm 0.44\%$	$34.97\% \pm 0.45\%$
	SR	$43.70\% \pm 0.56\%$	$\mathbf{55.87\% \pm 0.46\%}$	$\mathbf{26.60\% \pm 0.55\%}$	$\mathbf{38.28\% \pm 0.28\%}$
Alg. 2, $K = 3, \lambda = 0$	1-NN	$42.89\% \pm 0.43\%$	$53.13\% \pm 0.27\%$	$25.10\% \pm 0.48\%$	$34.52\% \pm 0.51\%$
	SR	$43.02\% \pm 0.11\%$	$55.74\% \pm 0.21\%$	$25.71\% \pm 0.41\%$	$37.68\% \pm 0.25\%$

The goal of this section is to prove that our proposed algorithm is not influenced by the choice of few-shot learning algorithm. We try different model heads after extracting the high-level features of the backbone. We select the Pre-trained Selection setting on ImageNet to demonstrate the viewpoint. The result is shown in Table 1. 1-NN means that we use a 1-NN algorithm based on cosine similarity to give the label of a test sample as the one with the nearest class centroid. SR means Softmax Regression on the high-level representation space. (*i.e.* Fine-tuning the classification layer in original backbone). The two methods represent for an easy realization of metric-based method and learning-based method. From Table 1 we show that the promotion of SR compared with 1-NN for all selection algorithms is rather stable in the same experiment setting and our algorithm is model-agnostic. Moreover, comparing 5-shot with 20-shot, we find that when the shot number is increasing, the margin of our algorithm and the baselines is shrinking when using SR as model head, which shows that the effect of fine-tuning gradually surpasses the effect of class selection with the increase of the shot number, demonstrating that our algorithm performs much better on few-shot setting.

4.2. Effects of the Number of Novel Classes

Table 2. ImageNet: Pre-trained Selection, 10-way

Algorithm	**m=100, 20-shot**
Random	$84.33\% \pm 1.71\%$
DomSim	$85.78\% \pm 2.06\%$
Alg. 1, $K = 10, \lambda = 0$	$88.36\% \pm 1.15\%$
Alg. 2, $K = 3, \lambda = 0$	$87.62\% \pm 1.38\%$
Alg. 2, $K = 10, \lambda = 0$	$88.02\% \pm 1.58\%$
Alg. 4, $K = 10, \lambda = 0.2$	$\mathbf{88.52\% \pm 1.88\%}$

In this section, we show the experiment result for 10-way 20-shot setting with $m = 100$ with 1-NN head in Table 2. We could draw three conclusions: First, in 10-way setting, our algorithm promotes about 4.19% compared with Random Selection, which is at the same level with 100-way 20-shot setting shown in Table 1, demonstrating the effectiveness of our proposed algorithm in different number of novel classes. Second, we find that we need to increase K compared with 100-way 20-shot setting as the number of base classes far exceeds the number of novel classes, thus we could provide more similar base classes for each novel class to improve the performance. Third, compared with $\lambda > 0$ and $\lambda = 0$ case, we show that diversity may be helpful when the number of base classes is much larger than the number of novel classes. In this setting diversity brings about a promotion of 0.5%.

4.3. Cold Start Selection on Caltech and CUB dataset

Table 3. Cold Start Selection, 100-way

Algorithm	**m=100, 5-shot, Caltech**	**m=100, 5-shot, CUB**
Random	$18.46\% \pm 1.19\%$	$45.31\% \pm 1.32\%$
DomSim	$27.32\% \pm 0.82\%$	$51.72\% \pm 1.24\%$
Alg. 2, $K = 1, \lambda = 0$	$27.59\% \pm 0.76\%$	$53.48\% \pm 1.19\%$
Alg. 2, $K = 3, \lambda = 0$	$\mathbf{29.33\% \pm 0.69\%}$	$\mathbf{53.56\% \pm 1.34\%}$
Alg. 2, $K = 5, \lambda = 0$	$28.83\% \pm 0.60\%$	$53.33\% \pm 1.18\%$
Pre-trained (Upperbound)	$29.65\% \pm 0.82\%$	$55.41\% \pm 1.25\%$

In this section, we also test cold start selection on Caltech256 and CUB-200-2011 as Table 3. All our algorithms use a mechanism of 6-12-25-50-100. The conclusion is the same as the original paper and there is nothing to discuss more about the results.

5. Detailed Experiment Settings in Training Phase

For all experiments, when training the base model, we use a standard ResNet-18 structure. The output dimension of the high-level feature is 512. The preprocessing step of the images is the same as original ResNet-18 paper. The base model is trained for 120 epoches, the learning rate is set to 0.1 for Epoch 1 to Epoch 25, 0.01 for Epoch 25 to Epoch 50, 0.001 for Epoch 50 to Epoch 80, 0.0001 for Epoch 80 to Epoch 105 and 0.00001 for Epoch 105 to Epoch 120. A weight decay with hyperparameter 0.0005 is used. We use a momentum SGD and the momentum coefficient is set to 0.9. The batch size is set to 64. We train the whole base model on 8*Nvidia Tesla V100. For each base model, the training time is about 4 hours and for each experiment setting this training process is repeated for 10 times, the total training hours for each experiment setting (*i.e.* each result number in the result tables) is about 40 hours. (The cold start problem may spend much longer time, about 75 hours per experiment setting). The main framework of the training process is based on Tensorflow, and the selection algorithm is based on C++11 for speed-up.

References

[1] A. A. Ageev and M. I. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. Society for Industrial and Applied Mathematics, 2014.

[3] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.